Programmer's Reference

Publication Number 54710-97003
Fifth Edition, October 1993

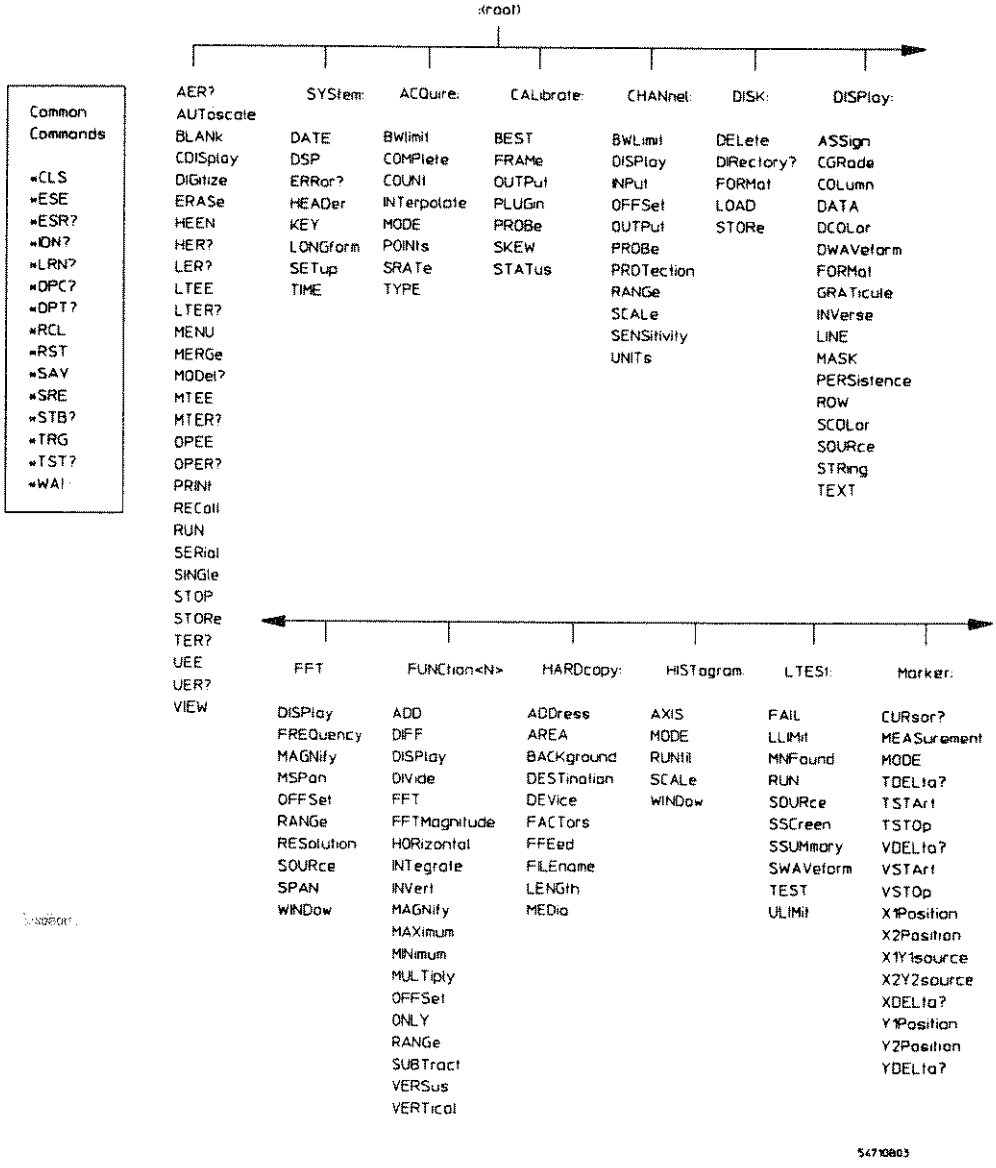
This reference applies directly to firmware revision code 3.XX.

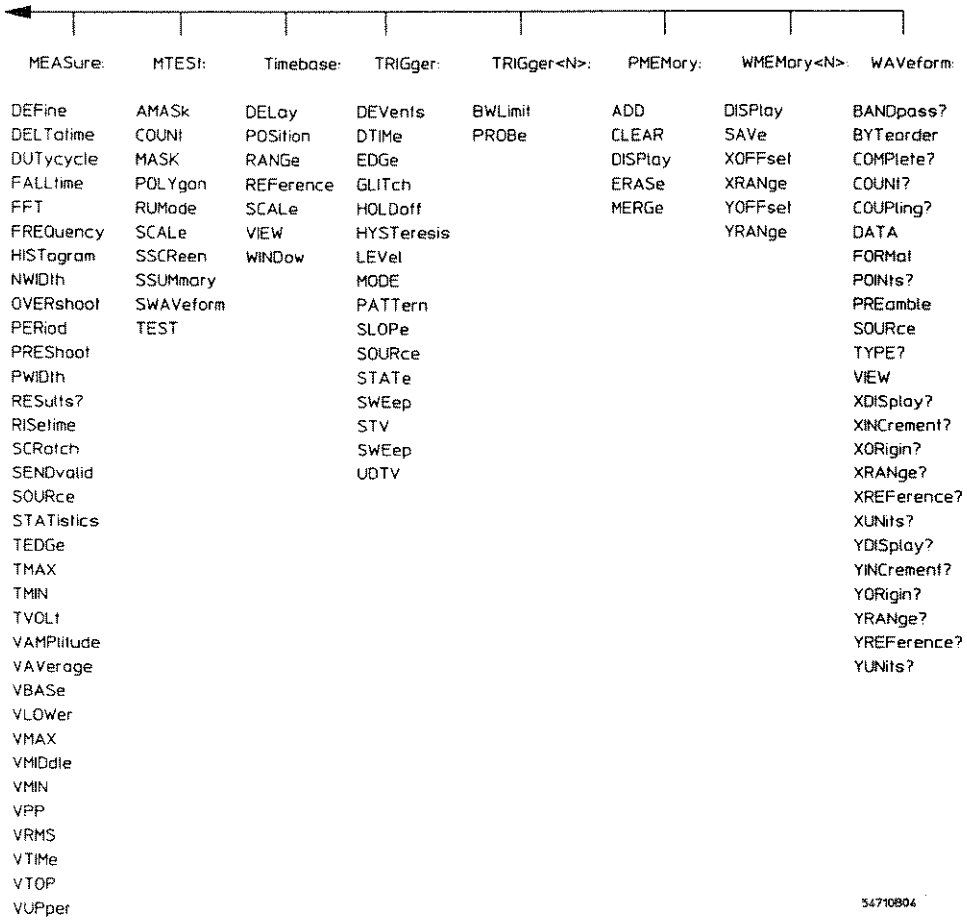
For Safety information, Warranties, and Regulatory information, see the pages behind the index.

©Copyright Hewlett-Packard Company1992, 1993
All Rights Reserved

HP 54710 and HP 54720 Oscilloscopes

The HP 54710 and HP 54720 Oscilloscope Programming Command Set




MEASure:
MTESt:
Timebase:
TRIGger:
TRIGger<N>:
PMEMory:
WMEMory<N>:
WAVeform:
DEFine
DELTatime
DUTycycle
FALLtime
FFT
FREQuency
HISTogram
NWIDth
OVERshoot
PERiod
PREShoot
PWIDth
RESults?
RISetime
SCRatch
SENDvalid
SOURce
STATistics
TEDGe
TMAX
TMIN
TVOLt
VAMPlitude
VAVerage
VBASe
VLOWer
VMAX
VMIDdle
VMIN
VPP
VRMS
VTIMe
VTOP
VUPper
AMASk
COUNt
MASK
POLYgon
RUMode
SCALe
SSCReen
SSUMmary
SWAVeform
TEST
DELay
POSition
RANGe
REFerence
SCALe
VIEW
WINDow
DEVents
DTIMe
EDGe
GLITch
HOLDoff
HYSTeresis
LEVel
MODE
PATTern
SLOPe
SOURce
STATe
SWEep
STV
SWEep
UDTV
BWLimit
PROBe
ADD
CLEAR
DISPlay
ERASe
MERGe
DISPlay
SAVe
XOFFset
XRANge
YOFFset
YRANge
BANDpass?
BYTeorder
COMPlete?
COUNt?
COUPling?
DATA
FORMat
POINts?
PREamble
SOURce
TYPE?
VIEW
XDISplay?
XINCrement?
XORigin?
XRANge?
XREFerence?
XUNits?
YDISplay?
YINCrement?
YORigin?
YRANge?
YREFerence?
YUNits?
54710804

In This Book

This book is your guide to programming the HP 54710 and HP 54720 Digitizing Oscilloscopes using the HP-IB command set.

Part One, "Introduction to Programming the HP 54710/HP 54720 Oscilloscopes," gives you the conceptual information needed to start programming the oscilloscope. This part includes information about basic program communications, interface, syntax, data types, and status reporting. It also has a set of sample programs that show you some typical applications.

Part Two, "HP 54710/HP 54720 HP-IB Command Reference," describes all the commands used to program the oscilloscope. Each chapter lists the commands that belong to an individual subsystem, and explains the function of each command.

Contents

2 Interface Functions

3 Message Communication and System Functions

4 Status Reporting

5 Programming Syntax

6 Programming Conventions

7 Example Programs

Part 2 HP 54710/HP 54720 HP-IB Command Reference

8 Common Commands

9 Root Level Commands

16 Function Commands

17 Hardcopy Commands

18 Marker Commands

19 Measure Commands

27 Limit Test Commands

28 Mask Test Commands

Part 2

HP 54710/HP 54720 HP-IB Command Reference

8

Common Commands

Common Commands

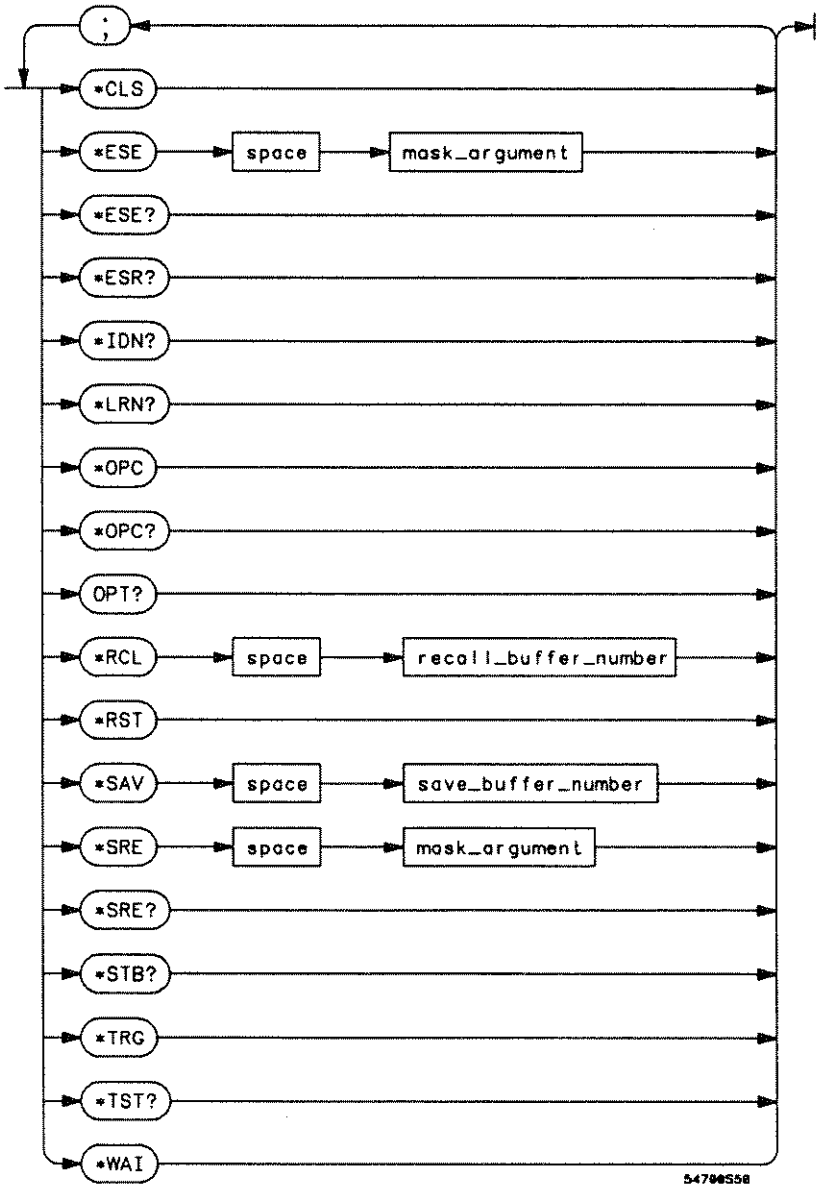
Common commands are defined by the IEEE 488.2 standard. They control generic device functions which could be common among many different types of instruments. Common commands can be received and processed by the instrument whether they are sent over the HP-IB as separate program messages or within other program messages.

The following common commands and queries are implemented in this oscilloscope:

- *CLS (Clear Status),
- *ESE (Event Status Enable),
- *ESR? (Event Status Register),
- *IDN? (Identification Number),
- *LRN? (Learn),
- *OPC (Operation Complete),
- *OPT? (Option),
- *RCL (Recall),
- *RST (Reset),
- *SAV (Save),
- *SRE (Service Request Enable),
- *STB? (Status Byte),
- *TRG (Trigger),
- *TST? (Test), and
- *WAI (Wait-to-Continue).

Figure 8-1 is the syntax diagram for the common commands.

Figure 8-1



Common Commands Syntax Diagram

Receiving Common Commands

Common commands can be received and processed by the oscilloscope whether they are sent over the HP-IB as separate program messages or within other program messages. If a subsystem is currently selected and a common command is received by the oscilloscope, the oscilloscope remains in the selected subsystem. For example, if the program message

```
"ACQUIRE:TYPE AVERAGE;*CLS;COUNT 1024"
```

is received by the oscilloscope, the oscilloscope sets the acquire type, clears the status information, and then sets the acquire count without leaving the selected subsystem.

Status Registers

The following two status registers used by common commands have an enable (mask) register. By setting bits in the enable register, the status information can be selected for use. Refer to the chapter, "Status Reporting," for a complete discussion of status.

Table 8-1 Status Registers

Status Register	Enable Register
Event Status Register	Event Status Enable Register
Status Byte Register	Service Request Enable Register

*CLS (Clear Status)

Command

```
*CLS
```

The *CLS command clears the status data structures including the device defined error queue. It also clears the Request-for-OPC flag.

Example

The following example clears the status data structures of the oscilloscope.

```
10  OUTPUT 707;"*CLS"
20  END
```

See Also

Refer to the "Status Reporting" chapter for a complete discussion of status.

*ESE (Event Status Enable)

Command

```
*ESE <mask>
```

The *ESE command sets the Standard Event Status Enable Register bits.

`<mask>` An integer, 0 to 255, representing a mask value for the bits to be enabled in the Standard Event Status Register as shown in table 8-2.

Example

The following example enables the User Request (URQ) bit of the Standard Event Status Enable Register. When this bit is enabled and a front-panel key is pressed, the Event Summary bit (ESB) in the Status Byte Register is also set.

```
10  OUTPUT 707;"*ESE 64"
20  END
```

Query

```
*ESE?
```

The *ESE query returns the current contents of the Standard Event Status Enable Register.

Returned Format

```
<mask><NL>
```

`<mask>` An integer, +0 to +255 (plus sign also returned), representing a mask value for the bits enabled in the Standard Event Status Register as shown in table 8-2.

Example

The following example places the current contents of the Standard Event Status Enable Register in the numeric variable, Event. The value of the variable is printed on the controller's screen.

```
10  OUTPUT 707;"*ESE?"
20  ENTER 707;Event
30  PRINT Event
40  END
```

The Standard Event Status Enable Register contains a mask value for the bits to be enabled in the Standard Event Status Register. A 1 in the Standard Event Status Enable Register enables the corresponding bit in the Standard Event Status Register. A 0 in the enable register disables the corresponding bit.

Table 8-2 Standard Event Status Enable Register Bits

Bit	Weight	Enables
7	128	PON - Power On
6	64	URQ - User Request
5	32	CME - Command Error
4	16	EXE - Execution Error
3	8	DDE - Device Dependent Error
2	4	QYE - Query Error
1	2	RQC - Request Control
0	1	OPC - Operation Complete

See Also

Refer to the chapter, "Status Reporting," for a complete discussion of status.

*ESR? (Event Status Register)

Query

```
*ESR?
```

The *ESR query returns the contents of the Standard Event Status Register. Reading this register clears the Standard Event Status Register.

Returned Format

```
<status><NL>
```

`<status>` An integer, 0 to 255, representing the total bit weights of all bits that are high at the time you read the register.

Example

The following example places the current contents of the Standard Event Status Register in the numeric variable, Event, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*ESR?"
20  ENTER 707;Event
30  PRINT Event
40  END
```

Table 8-3 lists each bit in the Event Status Register and the corresponding bit weights.

Table 8-3 Standard Event Status Register Bits

Bit	Bit Weight	Bit Name	Condition
7	128	PON	1 = OFF to ON transition has occurred.
6	64	URQ	0 = no front-panel key has been pressed. 1 = a front-panel key has been pressed.
5	32	CME	0 = no command errors. 1 = a command error has been detected.
4	16	EXE	0 = no execution error. 1 = an execution error has been detected.
3	8	DDE	0 = no device-dependent errors. 1 = a device-dependent error has been detected.
2	4	QYE	0 = no query errors. 1 = a query error has been detected.
1	2	RQC	0 = request control - NOT used - always 0.
0	1	OPC	0 = operation is not complete. 1 = operation is complete.
	0 = False = Low		1 = True = High

*IDN? (Identification Number)

Query

```
*IDN?
```

The *IDN query returns the instrument model number, serial number, and software version by returning the following string:

```
"HEWLETT-PACKARD,54720A,<XXXXAYYYYY>,<Rev #>
```

<XXXXAYYYYY> This specifies the serial number of the instrument. The first four digits and letter are the serial prefix, which is the same for all identical instruments. The last five digits are the serial suffix, which is assigned sequentially, and is different for each instrument.

<Rev #> This specifies the software version of the instrument and is the revision number.

Returned Format

```
HEWLETT-PACKARD,54720A,XXXXAYYYYY,X.XX<NL>
```

Example

The following example places the instrument's identification number in the string variable, Identify$, then prints the identification number to the controller's screen.

```
10  DIM Identify$[50]          !dimension variable
20  OUTPUT 707;"*IDN?"
30  ENTER 707;Identify$
40  PRINT Identify$
50  END
```

*LRN? (Learn)

Query

```
*LRN?
```

The *LRN query returns a response message (learn string) that contains the instrument's current setup. The instrument's setup can be stored and sent back to the instrument at a later time.

Returned Format

```
:SYSTem:SETup <setup><NL>
```

`<setup>` This is a definite length arbitrary block response specifying the current instrument setup. The block size is subject to change with different firmware revisions.

Example

The following example places the instrument's current setup in the string variable, Current$.

```
10  DIM Current$[10000]              !Dimension variable
20  OUTPUT 707;"*LRN?"
30  ENTER 707 USING "-K";Current$
40  OUTPUT 707; Current$             !Sends entire string including
50                                   !command header
60  END
```

The *LRN query always returns :SYSTEM:SETUP as a prefix to the setup block. The SYSTEM:HEADER command has no effect on this response.

See Also

The SYSTEM:SETUP? command/query. When HEADers and LONGform are ON, it performs the same function as the *LRN query. Otherwise, *LRN and SETup are not interchangeable.

*OPC (Operation Complete)

Command

```
*OPC
```

The *OPC command sets the operation complete bit in the Standard Event Status Register when all pending device operations have finished.

Example

The following example sets the operation complete bit in the Standard Event Status Register when the DIGITIZE operation is complete.

```
10  OUTPUT 707;":DIGITIZE:CHANNEL1;*OPC"
20  END
```

Query

```
*OPC?
```

The *OPC query places an ASCII character "1" in the instrument's output queue when all pending selected device operations have finished.

Returned Format

```
1<NL>
```

Example

The following example places an ASCII character "1" in the instrument's output queue when the AUTOSCALE operation is complete. Then the value in the output queue is placed in the numeric variable "Complete."

```
10  OUTPUT 707;":AUTOSCALE;*OPC?"
20  ENTER 707;Complete
30  END
```

The *OPC query allows synchronization between the controller and the instrument by using the message available (MAV) bit in the Status Byte, or by reading the output queue. Unlike the *OPC command, the *OPC query does not effect the OPC Event bit in the Standard Event Status Register.

*OPT (Option)

Query

```
*OPT?
```

The OPT? query returns the model number and serial number of the mainframe and all installed plug-in options as in the following string:

```
<MF_MOD_NO> <XXXXAYYYYY>,<PI_MOD_NO> <XXXXAYYYYY>,...
```

Plug-in information is returned for each plug-in option installed in the mainframe. If plug-in options are not installed, a zero is returned.

Note that the length of the returned option string may increase in future releases as more options become available. Once implemented, the option name (delimited by a comma) will be appended to the end of the returned string.

`<MF_MOD_NO>` This is the HP model number of the oscilloscope mainframe.

`<PI_MOD_NO>` This is the HP model number of the plug-in.

`<XXXXAYYYYY>` This is the serial number of the mainframe or plug-in respectively. The first four digits and letter are the serial prefix which is the same for all identical units. The last five digits are the serial suffix which is assigned sequentially and is different for each unit.

Example

This example places all options into the string variable, Options$, then prints the option model and serial numbers to the controller's screen.

```
10  DIM Options$[100]
20  OUTPUT 707;"*OPT?"
30  ENTER 707;Options$
40  PRINT Options$
50  END
```

*RCL (Recall)

Command

```
*RCL <register>
```

The *RCL command restores the state of the instrument to the setup previously stored in the specified save/recall register. An instrument setup must have been stored previously in the specified register. Registers 0 through 9 are general-purpose registers and can be used by the *RCL command.

`<register>` An integer, 0 through 9, specifying the save/recall register that contains the instrument setup you want to recall.

Example

The following example restores the instrument to the instrument setup stored in register 3.

```
10  OUTPUT 707;"*RCL 3"
20  END
```

An error message appears on screen if nothing has been previously saved in the specified register.

*RST (Reset)

Command

```
*RST
```

The *RST command places the instrument in a known state. The following table lists the reset conditions as they relate to the instrument commands.

Example

The following example resets the instrument to a known state.

```
10  OUTPUT 707;"*RST"
20  END
```

Table 8-4 *RST Reset Conditions

Global	
Run/Stop	Run
Menu	Unchanged
Headers	Off
Longform	Off
Time base	
Scale	20 us/div
Position	0 s
Reference	Center
Windowing	Disabled
Window scale	10 us
Window position	0 s

Trigger	
Mode	Edge
Level	0 V
Sweep	Auto
Hysteresis	Normal
Holdoff	140 ns
Edge source	First internal trigger; if none, first external trigger
Glitch source	First internal trigger; if none, first external trigger
Slope	Positive
Glitch polarity	Negative
Glitch width	20 ns
Pattern	When entered
Pattern definition	All don't care
Pattern GT	20 ns
Pattern LT	50 ns
Pattern range	20 to 50 ns
Delay time	30 ns
Acquisition	
Record length	Automatic
Sampling mode	Equivalent time
Completion	Off
Averaging	Off
Number of averages	16
Digital BW limit	Off
Interpolate	On
Sampling rate	Automatic

Display	
Persistence	Variable
Persistence time	Minimum
Draw waveforms	Fast
Graticule	Grid
Intensity	20
Graphs	1
Channel position	Channel 1 upper
	Channel 2 upper
	Channel 3 upper
	Channel 4 upper
Colors	Default
Marker	
Mode	Measurement
Readout	Off
Waveform + source	First available channel
Waveform + position	0 s
Waveform X source	First available channel
Waveform X position	0 s
X1, Y1 source	First available channel
X1 position	0 s
Y1 position	0 V
X2, Y2 source	First available channel
X2 position	0 s
Y2 position	0 V

Define measure	
Thresholds-percent	10%, 50%, 90%
Thresholds-volts	0.0, 1.6, 5.0
Top-base	Calculated
Statistics	Off
Delta time	
Start edge	Rising
Start edge number	1
Start edge threshold	Middle
Stop edge	Falling
Stop edge number	1
Stop edge threshold	Middle
Waveform	
Pixel memory state	Off
Waveform save format	text
Byte order	MSB first
Waveform source	Memory 1
Memory type	Waveform
Memory Display	Off
Y Scale	Unchanged
Y Offset	Unchanged
X Scale	Unchanged
X Position	Unchanged

Math	
Function	f1
Function state	Off
Operator	Magnify
Operand 1	First available channel or memory 1
Operand 2	First available channel or memory 1
Vertical scaling	Track Source
Horizontal Scaling	Track Source
Channel	
Display	On
Scale	1 V/div or maximum
Offset	0 V
54711A Sensitivity	Default
Probe atten units	Ratio
Probe attenuation	Unchanged
Units	Volts
External offset	0.0
External gain	1.0
Input	dc 1 Mohm if available or dc 50 ohm
Utility	
CRT pattern	Off
Light output	Off
Color purity	Off
Color ID	Current color ID (0)
Disk	
operation	Unchanged

*SAV (Save)

Command

```
*SAV <register>
```

The *SAV command stores the current state of the instrument in a save register.

`<register>` An integer, 0 through 9, specifying the save/recall register in which to save the current instrument setup.

Example

The following example stores the current instrument setup to register 3.

```
10  OUTPUT 707;"*SAV 3"
20  END
```

*SRE (Service Request Enable)

Command

```
*SRE <mask>
```

The *SRE command sets the Service Request Enable Register bits.

`<mask>` An integer, 0 to 255, representing a mask value for the bits to be enabled in the Service Request Enable Register as shown in table 8-5.

Example

The following example enables a service request to be generated when a message is available in the output queue. When a message is available, the MAV bit is high.

```
10  OUTPUT 707;"*SRE 16"
20  END
```

Query

```
*SRE?
```

The *SRE query returns the current contents of the Service Request Enable Register.

Returned Format

```
<mask><NL>
```

`<mask>` An integer, 0 to 255, representing a mask value for the bits enabled in the Service Request Enable Register.

Example

The following example places the current contents of the Service Request Enable Register in the numeric variable, Value, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*SRE?"
20  ENTER 707;Value
30  PRINT Value
40  END
```

The Service Request Enable Register contains a mask value for the bits to be enabled in the Status Byte Register. A 1 in the Service Request Enable Register enables the corresponding bit in the Status Byte Register. A 0 disables the bit.

Table 8-5 Service Request Enable Register Bits

Bit	Weight	Enables
7	128	OPER - Operation Status Register
6	64	Not Used
5	32	ESB - Event Status Bit
4	16	MAV - Message Available
3	8	Not Used
2	4	MSG - Message
1	2	USR - User Event Register
0	1	TRG - Trigger

*STB? (Status Byte)

Query `*STB?`

The *STB query returns the current contents of the Status Byte including the Master Summary Status (MSS) bit.

Returned Format `<value><NL>`

`<value>` An integer, 0 to 255, representing a mask value for the bits enabled in the Status Byte.

Example The following example reads the contents of the Status Byte into the numeric variable, Value, then prints the value of the variable to the controller's screen.

```
10  OUTPUT 707;"*STB?"
20  ENTER 707;Value
30  PRINT Value
40  END
```

In response to a serial poll (SPOLL), Request Service (RQS) is reported on bit 6 of the status byte. Otherwise, the Master Summary Status bit (MSS) is reported on bit 6. MSS is the inclusive OR of the bitwise combination (excluding bit 6) of the Status Byte Register and the Service Request Enable Register. The MSS message indicates that the instrument has at least one reason for requesting service.

Table 8-6 **Status Byte Register Bits**

Bit	Bit Weight	Bit Name	Condition
7	128	OPER	0 = no enabled operation status conditions have occurred 1 = an enabled operation status condition has occurred
6	64	RQS/MSS	0 = instrument has no reason for service 1 = instrument is requesting service
5	32	ESB	0 = no event status conditions have occurred 1 = an enabled event status condition occurred
4	16	MAV	0 = no output messages are ready 1 = an output message is ready
3	8	---	0 = not used
2	4	MSG	0 = no message has been displayed 1 = message has been displayed
1	2	USR	0 = no enabled user event conditions have occurred 1 = an enabled user event condition has occurred
0	1	TRG	0 = no trigger has occurred 1 = a trigger occurred
	0 = False = Low		1 = True = High

*TRG (Trigger)

Command

```
*TRG
```

The *TRG command has the same effect as the Group Execute Trigger message (GET) or RUN command. It acquires data for the active waveform display according to the current settings.

Example

The following example starts the data acquisition for the active waveform display according to the current settings.

```
10  OUTPUT 707;"*TRG"
20  END
```

*TST? (Test)

Query

```
*TST?
```

The *TST query causes the instrument to perform a self-test and places a response in the output queue indicating whether or not the self-test completed without any detected errors. A zero indicates that the test passed and a non-zero indicates the test failed.

> All front-panel inputs must be disconnected before sending this command.

Returned Format

```
<result><NL>
```

`<result>` 0 for pass; non-zero for fail.

Example

The following example performs a self-test on the instrument and places the results in the numeric variable, Results. Then the program prints the results to the controller's screen.

```
10  OUTPUT 707;"*TST?"
20  ENTER 707;Results
30  PRINT Results
40  END
```

If a test fails, refer to the troubleshooting section of the service guide.

*WAI (Wait-to-Continue)

Command `*WAI`

The *WAI command prevents the instrument from executing any further commands or queries until all curently executing commands are completed.

9

Root Level Commands

Root Level Commands

Root level commands control many of the basic operations of the oscilloscope that can be selected by pressing the labeled keys on the front panel. These commands are always recognized by the parser if they are prefixed with a colon, regardless of the current tree position. After executing a root level command, the parser is positioned at the root of the command tree.

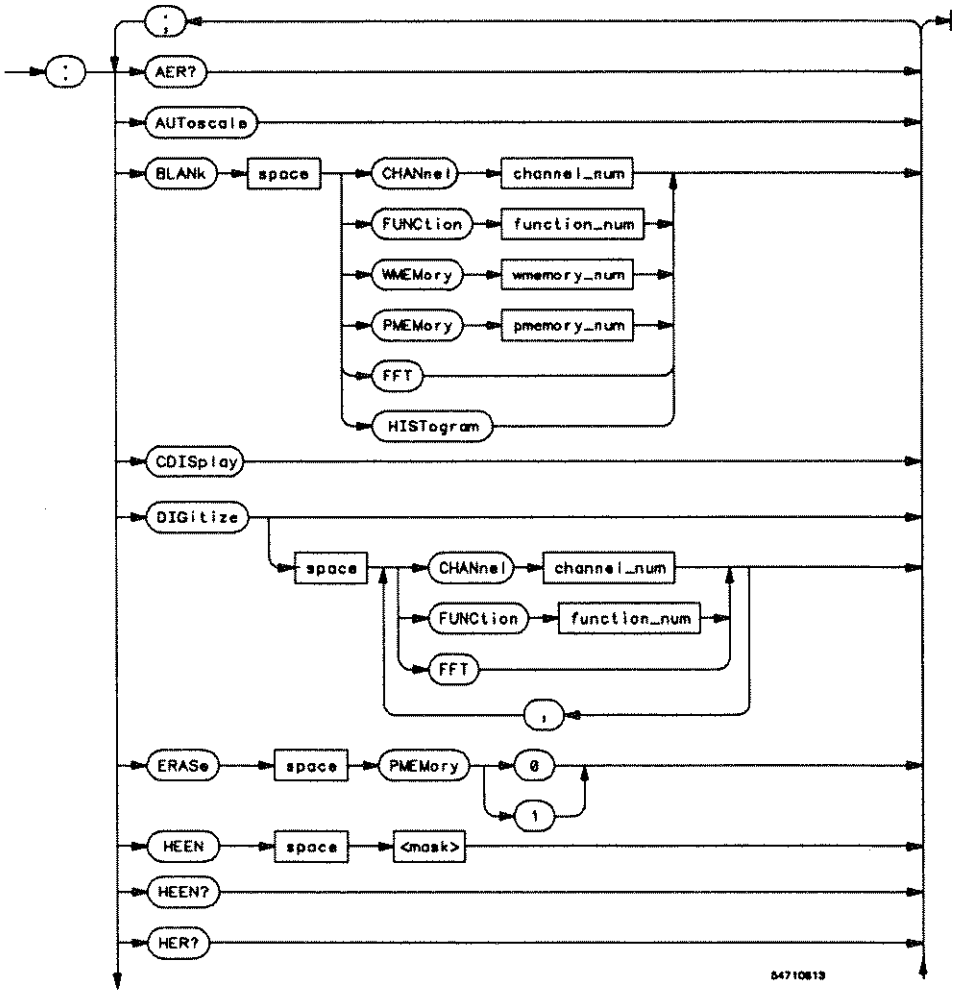
The following root level commands and queries are implemented in this oscilloscope:

- AER (Arm Event Register)?,
- AUToscale,
- BLANk,
- CDISplay,
- DIGitize,
- ERASe,
- HEEN (Histogram Event Enable),
- HER ? (Histogram Event Register),
- LER ? (Local Event Register),
- LTEE (Limit Test Event Enable),
- LTER? (Limit Test Event Register),
- MENU,
- MERGe,
- MODel?,
- MTEE (Mask Test Event Enable),
- MTER? (Mask Test Event Register),
- OPEE (Operation Status Enable),
- OPER? (Operation Status Register),
- PRINt,
- RECall:SETup,
- RUN,
- SERial (Serial Number),
- SINGle,
- STOP,
- STORe:PMEM1,
- STORe:SETup,
- STORe:WAVEform,
- TER? (Trigger Event Register),
- UEE (User Event Enable),

- UER? (User Event Register), and
- VIEW.

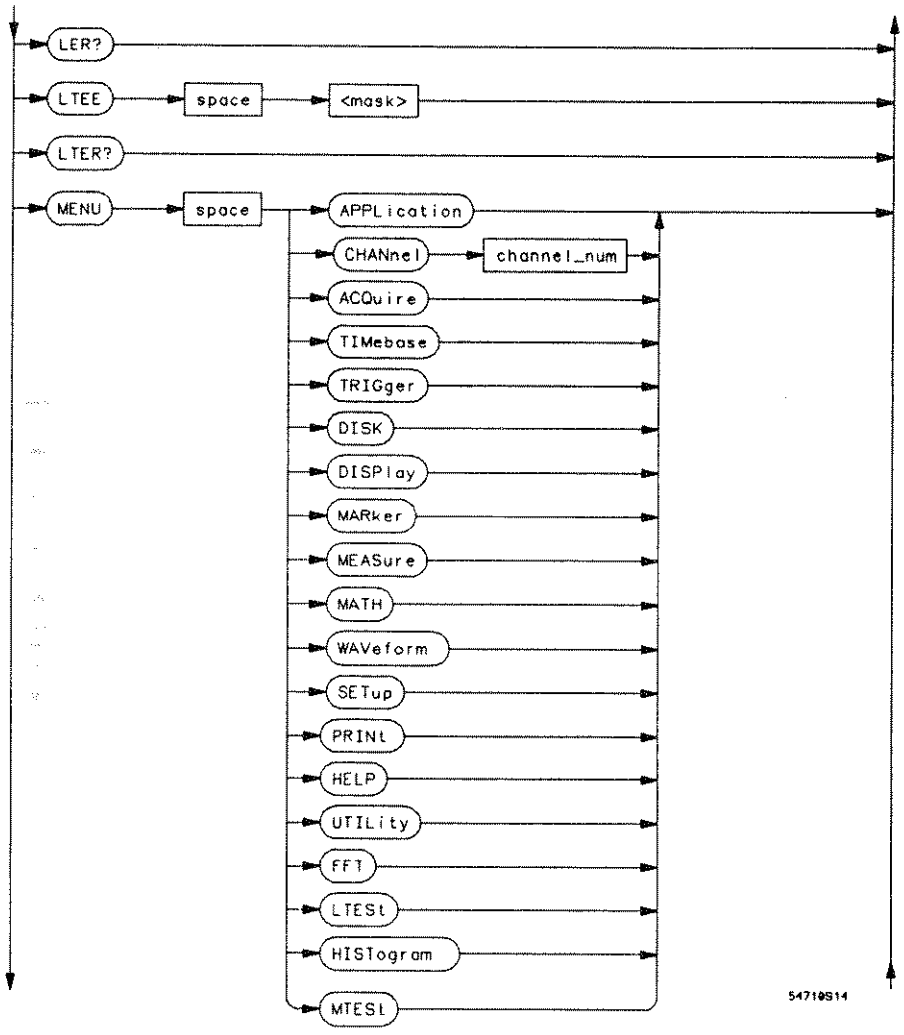
Figure 9-1 is the syntax diagram for the root level commands.

Figure 9-1



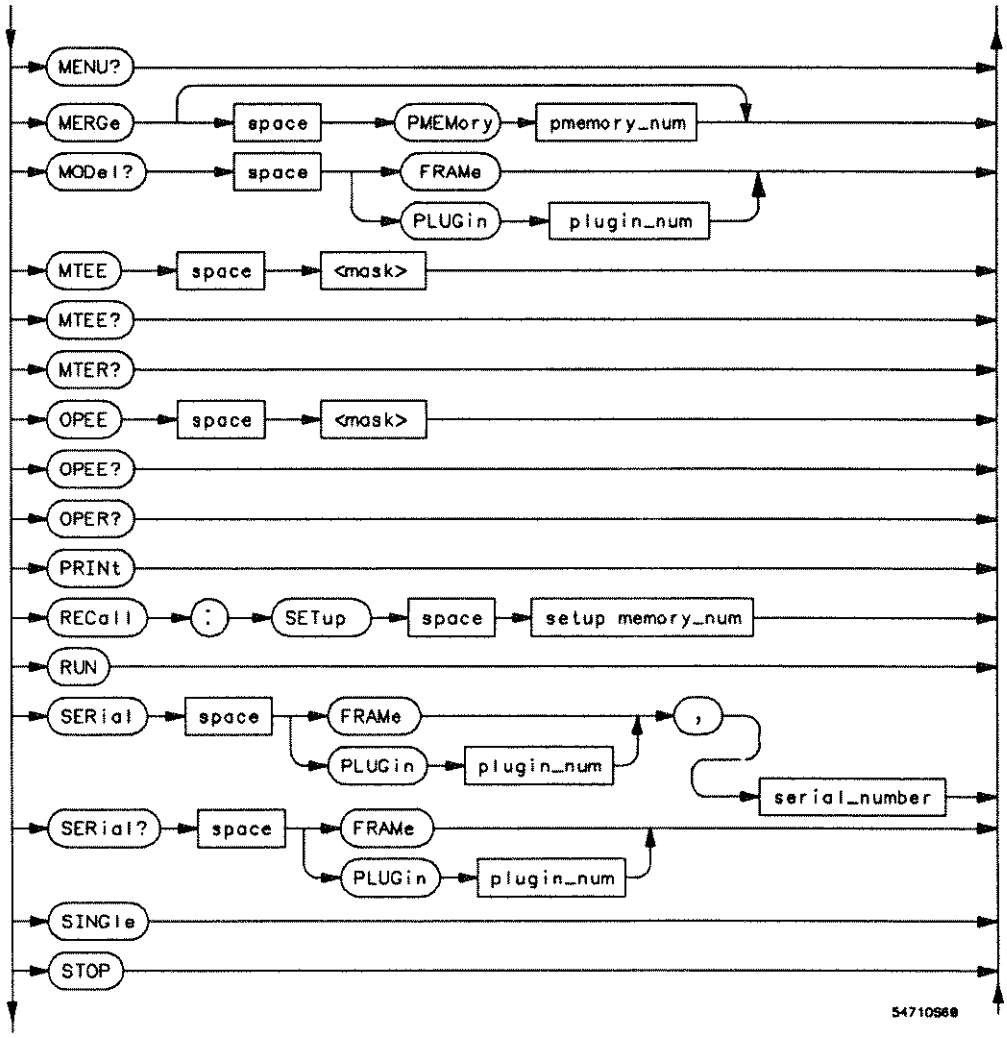
Root Level Syntax Diagram

Figure 9-1 (continued)



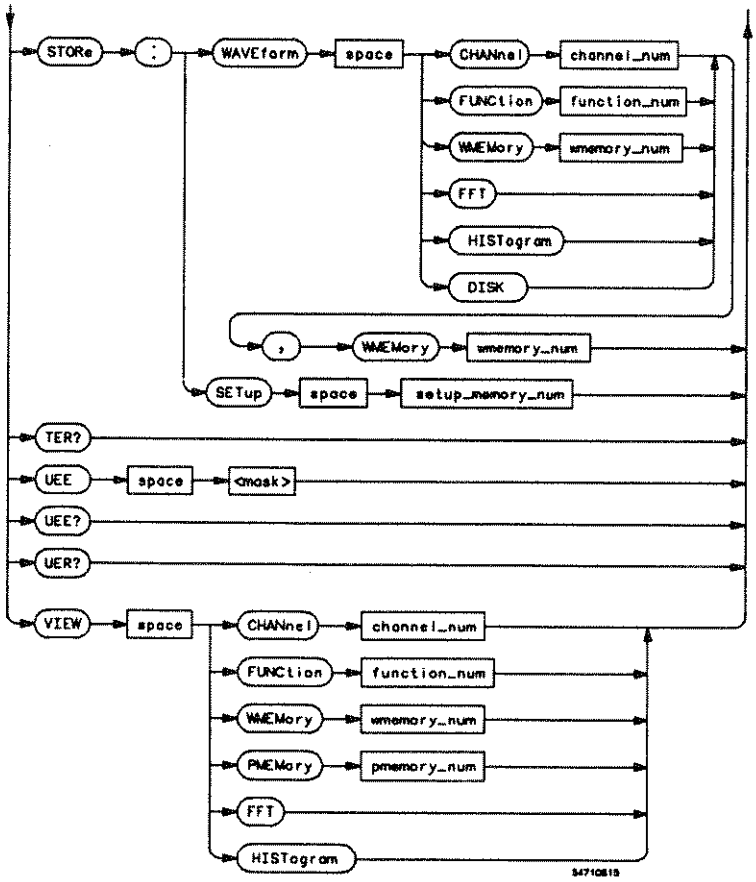
Root Level Syntax Diagram (continued)

Figure 9-1 (continued)



Root Level Syntax Diagram (continued)

Figure 9-1 (continued)



channel_num	1, 2, 3, or 4
function_num	1 or 2.
wmemory_num	1, 2, 3, or 4.
pmemory_num	1.
plugin_num	1, 2, 3, or 4.
setup memory num	0 through 9.

Root Level Syntax Diagram (continued)

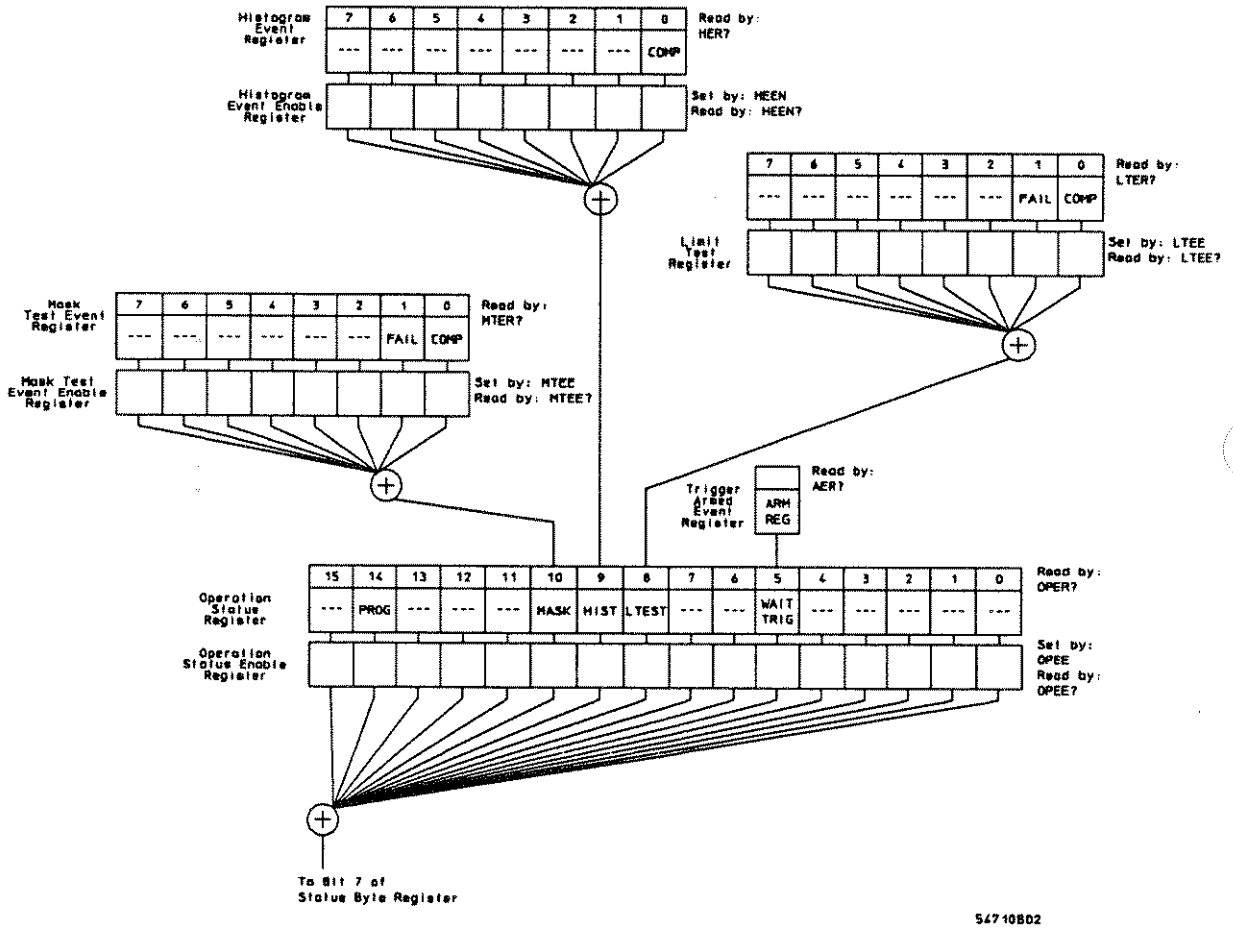
Status Reporting Data Structures

Figure 9-2 shows the different status reporting data structures and how they work together. To make it possible for any of the Standard Event Status Register bits to generate a summary bit, the bits must be enabled. These bits are enabled by using the *ESE common command to set the corresponding bit in the Standard Event Status Enable Register.

To generate a service request (SRQ) interrupt to an external controller, at least one bit in the Status Byte Register must be enabled. These bits are enabled by using the *SRE common command to set the corresponding bit in the Service Request Enable Register. These enabled bits can then set RQS and MSS (bit 6) in the Status Byte Register.

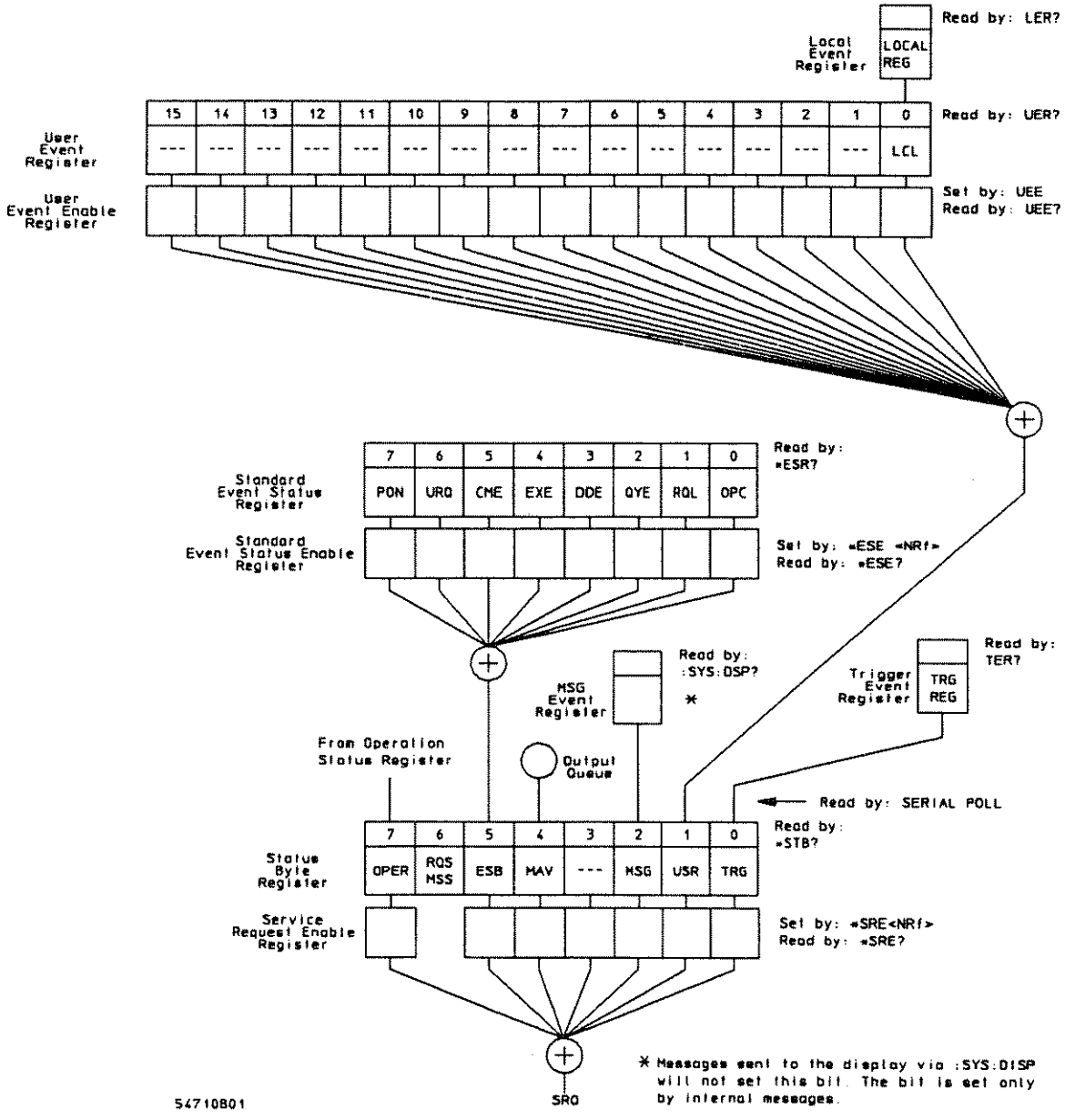
Various root level commands, documented in this chapter, are used to query and set various registers within the register set.

Figure 9-2



Status Reporting Data Structure

Figure 9-2 (continued)



Status Reporting Data Structure

AER? (Arm Event Register)

Query

```
:AER?
```

The AER? query reads the Arm Event Register and returns 1 or 0. After the Arm Event Register is read, the register is cleared. The returned value 1 indicates a trigger has occurred and 0 indicates a trigger has not occurred.

> The bit will only be set on the first trigger armed after the *CLS command is sent to the oscilloscope.

Once this bit is set, it is cleared only by reading the Status byte or by sending a *CLS command.

Returned Format

```
[:AER] {1 | 0}
```

AUToscale

Command `:AUToscale`

This command causes the instrument to evaluate all input signals and to find the optimum conditions for displaying the signal. It searches each of the installed channels for input signals and shuts off channels on which no signal is found. It adjusts the vertical gain and offset for each channel that has a signal, and sets the time base on the lowest numbered input channel with a signal. The trigger is found by searching each module from left-to-right, trying external trigger inputs before internal trigger inputs, until a trigger signal is detected. If signals cannot be found on any vertical input, the instrument is returned to its former state.

Autoscale sets the following conditions:

- Channel Display, Scale, and Offset,
- Trigger Sweep, Mode Edge, Source, Level, Slope, Hysteresis, and Holdoff,
- Acquisition Sampling rate and Record length,
- Time base Scale and Position,
- Marker mode set to measurement, and
- Resets completion criteria.

Autoscale turns off the following items:

- Measurements on sources that are turned off,
- Functions,
- Windows,
- Memories,
- Measurement limit test,
- Mask Test, and
- Histograms.

No other controls are affected by Autoscale.

Example

The following example automatically scales the oscilloscope for the input signal.

```
10  OUTPUT 707;":AUTOSCALE"
20  END
```

BLANk

Command

```
:BLANK {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | PMEMory<number> | FFT |
HISTogram}
```

This command turns off an active channel, function, waveform memory, pixel memory, FFT, or histogram. The VIEW command turns them on.

`<number>` For channels: 1, 2, 3, or 4.
For functions: 1 or 2.
For waveform memories (WMEMory): 1, 2, 3, or 4.
For pixel memory (PMEMory): 1.

Example

The following example turns off channel 1.

```
10  OUTPUT 707;":BLANK CHANNEL1"
20  END
```

CDISplay

Command

```
:CDISplay
```

The CDISplay command clears the display and resets all associated measurements. If the oscilloscope is in the stopped mode, all data that are currently displayed are erased. If the oscilloscope is running, all data in active channels and functions are erased; however, new data are displayed on the next acquisition. Waveform and pixel memories are not erased. Sending this command is the same as pressing the front panel Clear Display key, or sending :ERASE PMEMORY0.

Example

The following example uses the CDISplay command to clear the oscilloscope screen.

```
10  OUTPUT 707;":CDISPLAY"
20  END
```

DIGitize

Command

```
:DIGitize [<waveform_name>] [,<waveform_name>]
```

The DIGitize command invokes a special mode of data acquisition that is more efficient than using the RUN command. This command initializes the selected channels, functions, or FFT to "unacquired," and then acquires them according to the current instrument settings. When all signals are completely acquired, the instrument is placed in the stopped state. The waveform completion criteria is set with the ":ACQuire:COMPlete," command.

If channel, function, or FFT parameters are specified, then these are the only waveforms acquired. To speed acquisition, these waveforms are not displayed and their display state indicates "off." Subsequent to the digitize operation, the display of the acquired waveforms may be turned on for viewing, if desired. Other sources are turned off and their data are invalidated.

> Even though digitized waveforms are not displayed, the full range of measurement and math operators may be performed on them.

If the DIGitize command is issued without parameters, the digitize operation is performed on the channels, functions, and FFT that were acquired with a previous digitize, run, or single operation. In this case, the display state of the acquired waveforms is not changed. Because the command executes more quickly without parameters, this form of the command is useful for repetitive measurement sequences. Also, this mode can be used if it is desirable to view the digitize results because the display state of the digitized waveforms is not affected.

The data acquired with the DIGitize command are placed in the normal channel, function, or FFT memories just as when acquired via the RUN command.

`<waveform_name>` CHANnel<number>, FUNCtion<number>, or FFT

`<number>` For channels: 1, 2, 3, or 4.
For functions: 1 or 2.

Example

The following example uses the digitize command to acquire data on channel 1 and function 2.

```
10  OUTPUT 707;":DIGITIZE CHANNEL1,FUNCTION2"
20  END
```

The Acquire subsystem commands are used to set up conditions, such as TYPE and COUNT for the next DIGITIZE command. The Waveform subsystem commands are used to determine how the data is transferred out of the instrument and how to interpret the data.

ERASe

Command

```
:ERASe [PMEMory<number>]
```

This command erases the specified pixel memory.

`<number>` 0 or 1.

Example

The following example clears the specified pixel memory and anything on the display from that pixel memory.

```
10  OUTPUT 707;":ERASE PMEMORY1"
20  END
```

Erasing pixel memory 0 is a special case, depending on whether the oscilloscope is in the stopped mode or is running. If the oscilloscope is in the stopped mode, all data that are currently displayed are erased. If the oscilloscope is running, all data in active channels and functions are erased; however, new data are displayed on the next acquisition. Waveform and pixel memories are not erased. In addition, erasing pixel memory 0 is the same as pressing the CLEAR DISPLAY front-panel key. It clears the display and resets all associated measurements.

Erasing pixel memory 1 clears the static pixel memory and anything on the display from that pixel memory. Pixel memory 1 is the pixel memory accessible from the front-panel Waveform menu.

HEEN

Command

`HEEN <mask>`

This command sets a mask into the Histogram Event Enable register. A 1 in a bit position enables the corresponding bit in the Histogram Event register to set bit 9 in the Operation Status register.

`<mask>` The decimal weight of the enabled bits. Only bit 0 is used at this time; so to enable the COMP bit, set bit 0 to 1 with the HEEN command; otherwise, set bit 0 to 0.

Query

`HEEN?`

The query returns the current decimal value in the Histogram Event Enable register.

Returned Format

`[HEEN] <mask><NL>`

HER?

Query

`HER?`

This query returns the current value of the Histogram Event register as a decimal number.

Bit 0 (COMP) of the Histogram Event Register is set when the Histogram completes. The Histogram completion criteria are set by the HISTogram:RUNTil command.

Returned Format

`[HER?] <value><NL>`

LER? (Local Event Register)

Query

```
:LER?
```

This query reads the Local (LCL) Event Register. A "1" is returned if a remote-to-local transition has taken place due to the front-panel LOCAL key being pressed. A "0" is returned if a remote-to-local transition has not taken place.

Returned Format

```
[:LER] {1 | 0}<NL>
```

Example

The following example checks to see if a remote-to-local transition has taken place and places the result in the string variable, Answer$, and then it prints the result to the controller's screen.

```
10  DIM Answer$[50]           !Dimension variable
20  OUTPUT 707;":LER?"
30  ENTER 707;Answer$
40  PRINT Answer$
50  END
```

After the LCL Event Register is read, it is cleared.

Once this bit is set, it can only be cleared by reading the Status Byte, reading the register with the LER query, or sending a *CLS common command.

LTEE

Command

```
LTEE <mask>
```

This command sets a mask into the Limit Test Event Enable register. A 1 in a bit position enables the corresponding bit in the Limit Event register to set bit 8 in the Operation Status register.

`<mask>` The decimal weight of the enabled bits. Only bits 0 and 1 of the Limit Test Event Register are used at this time, so the useful mask values are given by the following table:

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

Query

```
LTEE?
```

The query returns the current decimal value in the Limit Test Event Enable register.

Returned Format

```
[LTEE] <mask><NL>
```

LTER?

Query

`LTER?`

This query returns the current value of the Limit Test Event register as a decimal number.

Bit 0 (COMP) of the Limit Test Event Register is set when the Limit Test completes. The Limit Test completion criteria are set by the LTESt:RUN command.

Bit 1 (FAIL) of the Limit Test Event Register is set when the Limit Test fails. Failure criteria for the Limit Test are defined by the LTESt:FAIL command.

Returned Format

`[LTER?] <value><NL>`

MENU

Command

```
:MENU { APPLication | CHANnel<number> | ACQuire |
TIMebase | TRIGger | DISK | DISPlay | MARKer |
MEASure | MATH | WAVeform | SETup | PRINt | HELP |
UTILity | FFT | LTESt | HISTogram  | MTESt }
```

This command selects one of the menus on the front panel.

Example

The following example displays the TIME BASE on the oscilloscope's screen.

```
10  OUTPUT 707;":MENU TIMEBASE"
20  END
```

Query

```
:MENU?
```

The MENU query returns the name of the current menu into a string.

Returned Format

```
[:MENU] { APPLication | CHANnel<number> | ACQuire | TIMebase |
TRIGger | DISK | DISPlay | MARKer | MEASure | MATH | WAVeform
| SETup | PRINt | HELP | UTILity | FFT | LTESt | HISTogram |
MTESt}
```

Example

The following example places the name of the currently displayed menu in the string variable, Answer$, and then prints the contents of the variable on the controller's screen.

```
10  DIM Answer$[50]           !Dimension variable
20  OUTPUT 707;":MENU?"
30  ENTER 707;Answer$
40  PRINT ANSWER$
50  END
```

MERGe

Command

```
:MERGe [PMEMory<number>]
```

This command stores the contents of the active display into the pixel memory. The stored data are essentially a copy of the waveform data that are presently displayed on screen. Display labels are also copied to pixel memory.

Since there is only one pixel memory in the instrument, the PMEMory<number> parameter is optional and PMEMory1 is the only choice if a parameter is sent.

Example

The following example stores the contents of the active display into pixel memory 1.

```
10  OUTPUT 707;":MERGE PMEMORY1"
20  END
```

MODel?

Query

```
:MODel? {FRAMe | PLUGin<number>|~KNOWN}
```

This query returns the HP model number for the frame or plug-in.

`<number>` The residing slot number of the plug-in.

Returned Format

A six-character alphanumeric model number within quotes. Output is determined by header and longform status as in table 9-1. A response of "~KNOWN" indicates that the identifier is corrupt.

Table 9-1 **Model? Returned Format**

HEADER		LONGFORM		RESPONSE
ON	OFF	ON	OFF	
	X		X	54720A
	X	X		54720A
X			X	:MOD 54720A
X		X		:MODEL 54720A

Example

The following example places the model number of the frame in a string variable, Model$, then prints the contents of the variable on the controller's screen.

```
10 Dim Model$[13]      !Dimension variable
20 OUTPUT 707;":Model? FRAME"
30 ENTER 707; Model$
40 PRINT MODEL$
50 END
```

MTEE

Command

`MTEE <mask>`

This command sets a mask into the Mask Event Enable register. A 1 in a bit position enables the corresponding bit in the Limit Event register to set bit 10 in the Operation Status register.

`<mask>` The decimal weight of the enabled bits. Only bits 0 and 1 of the Mask Test Event Register are used at this time, so the useful mask values are defined by the following table:

Enable	Mask Value
Block COMP and FAIL	0
Enable COMP, block FAIL	1
Enable FAIL, block COMP	2
Enable COMP and FAIL	3

Query

`MTEE?`

The query returns the current decimal value in the Mask Event Enable register.

Returned Format

`[MTEE] <mask><NL>`

MTER?

Query

`MTER?`

This query returns the current value of the Mask Event register as a decimal number.

Bit 0 (COMP) of the Mask Test Event Register is set when the Mask Test completes. The Mask Test completion criteria are set by the MTESt:RUMode command.

Bit 1 (FAIL) of the Mask Test Event Register is set when the Mask Test fails. This will occur whenever any sample is recorded within any polygon defined in the mask.

Returned Format

`[MTER?] <value><NL>`

OPEE

Command

```
:OPEE <mask>
```

This command sets a mask in the Operation Status Enable register. Each bit set to a 1 enables that bit to set bit 7 in the status byte register and potentially cause an SRQ to be generated. Bit 5, Wait for Trig (Trigger Armed), bit 8 Ltest (Limit Test), bit 9 Hist (Histogram), and bit 10 Mask (Mask Test) bits are used. Other bits are reserved.

`<mask>` The decimal weight of the enabled bits.

Query

```
OPEE?
```

The query returns the current value contained in the Operation Status Enable register as a decimal number.

Returned Format

```
[OPEE] <value><NL>
```

OPER?

Query

`OPER?`

The OPER? query returns the value contained in the Operation Status Register as a decimal number. This register hosts the WAIT TRIG bit (bit 5), the LTEST bit (bit 8), the HIST bit (bit 9), the MASK bit (bit 10), and the PROG bit (bit 14).

The WAIT TRIG bit is set by the Trigger Armed Event Register and indicates that the trigger is armed.

The LTEST bit is set when a limit test fails or is completed and sets the corresponding FAIL or COMP bits in the Limit Test Event Register.

The HIST bit is set when the COMP bit is set in the Histogram Event Register, indicating that the histogram measurement has satisfied the specified completion criteria.

The MASK bit is set when the Mask Test either fails specified conditions or satisfies its completion criteria, setting the corresponding FAIL or COMP bits in the Mask Test Event Register.

The PROG bit is reserved for future use.

Returned Format

`[OPER?] <value><NL>`

PRINt

Command

```
:PRINt
```

This command outputs a copy of the screen to a printer or other device destination specified in the hardcopy subsystem . The selection of the output and the printer can be specified using the HARDcopy subsystem commands.

Example

The following example outputs a copy of the screen to a printer, on the Centronics port, or to a disk file.

```
10  OUTPUT 707;":PRINT"
20  END
```

Example

If the destination is HP-IB, use the following commands after the PRINT command. This example assumes the printer HP-IB address is 1.

```
10 OUTPUT 707;":PRINT"
20 SEND 7;UNT UNL
30 SEND 7; LISTEN 1
40 SEND 7; TALK 7
50 SEND 7; DATA
60 END
```

After the print has completed, a "LOCAL 707" must be sent to release Local Lockout when using the example program.

RECall:SETup

Command

`:RECall:SETup <setup_num>`

This command recalls a setup. The SETup command, sent with the RECall command, recalls a setup from the internal memory that has been previously saved using the STORe:SETup command or saved from the front panel.

`<setup_num>` Setup number 0 through 9.

Examples

The following command recalls a setup from a setup memory that has been previously saved using the STORe:SETup command or saved from the front panel.

```
10 OUTPUT 707;":RECall:SETup 2"
20 END
```

RUN

Command

```
:RUN
```

This command places the instrument in the running state, in which waveforms are acquired according to the current settings. It enables acquisition cycles to run repetitively until the STOP command is received.

Example

The following example acquires data for the oscilloscope in a repetitive manner.

```
10  OUTPUT 707;":RUN"
20  END
```

SERial (Serial Number)

Command

```
:SERial {FRAMe | PLUGin<number>},<string>
```

This command sets the serial number for the oscilloscope frame or plug-in. The oscilloscope's serial number is entered at HP; therefore, setting the serial number is not normally required unless the instrument is serialized for a different application. The mainframe serial number is protected by the "Frame cal protected" switch on the rear panel. The plug-in serial numbers are protected by calibration protect switches inside the plug-in. The switches need to be in the unprotected position in order to program the serial numbers. The switch position is displayed in the UTILITY/System config front-panel menu.

This serial number is part of the string returned for the *IDN? query in the common commands.

`<string>` A ten-character alphanumeric serial number within quotes.

Example

The following example sets the serial number for the oscilloscope's frame to "1234A56789."

```
10  OUTPUT 707;":SERIAL FRAME,""1234A56789"""
20  END
```

Query

```
:SERial? {FRAMe | PLUGin <number>}
```

The query returns the current serial number string for the specified frame or plug-in.

Returned Format

```
[:SERial {FRAMe | PLUGin <number>}]
```

Example

```
10 Dim Serial$[50]            !Dimension variable
20 OUTPUT 707;":SERIAL? FRAME"
30 ENTER 707; Serial$
40 PRINT SERIAL$
50 END
```

SINGle

Command

```
:SINGle
```

The SINGle command causes the instrument to make a single acquisition when the next trigger event occurs.

The :STOP command should be sent before this command to ensure the instrument is not running. This will ensure that only a single trigger is accepted and 1 acquisition is acquired.

Example

The following example sets up the instrument to make a single acquisition when the next trigger event occurs.

```
10  OUTPUT 707;":SINGLE"
20  END
```

STOP

Command

```
:STOP
```

The STOP command causes the instrument to stop acquiring data for the active display. The RUN or SINGle command must be used to restart the acquisition.

Example

The following example stops the current data acquisition.

```
10  OUTPUT 707;":STOP"
20  END
```

STORe:SETup

Command

```
:STORe:SETup <setup_num>
```

The STORe:SETup command saves the current instrument setup in setup memories 0 through 9.

`<setup_num>` For setup memories 0 through 9.

STORe:WAVEform

Command

```
:STORe:WAVEform {CHANnel<number>|FUNCtion<number>|
WMEMory<number> | HISTogram | FFT},
{WMEMory<number>}
```

The :STORe:WAVeform command copies a channel, function, stored waveform, histogram, or FFT to a waveform memory. The first parameter specifies the source of the copy and can be any channel, function, waveform memory, histogram, or the FFT. The second parameter is the destination of the copy, and can be any waveform memory.

`<number>` For channels, 1, 2, 3, or 4
For functions, 1 or 2.
For waveform memories (WMEMory), 1, 2, 3, or 4.

Example

The following example copies channel 1 to waveform memory 3.

```
10  OUTPUT 707;":STORE:WAVEFORM CHANNEL1,WMEMORY3"
20  END
```

TER? (Trigger Event Register)

Query

```
:TER?
```

The TER query reads the Trigger Event Register. A "1" is returned if a trigger has occurred. A "0" is returned if a trigger has not occurred.

Returned Format

```
{1 | 0}<NL>
```

Example

The following example checks the current status of the Trigger Event Register and places that status in the string variable, Current$. Then it prints the contents of the variable to the controller's screen.

```
10  DIM Current$[50]          !Dimension variable
20  OUTPUT 707;":TER?"
30  ENTER 707;Current$
40  PRINT Current$
50  END
```

Once this bit is set, it can only be cleared by reading the Status Byte, reading the register with the :TER? query, or by sending a *CLS common command.

After the Trigger Event Register is read, it is cleared.

Althought TER is cleared by the :TER? query, it cannot be set again until after the *CLS command is sent.

UEE

Command `UEE <mask>`

This command sets a mask into the User Event Enable register. A 1 in a bit position enables the corresponding bit in the User Event register to set bit 1 in the Status Byte register and thereby potentially cause an SRQ to be generated. Only bit 0 of the User Event Register is used at this time; all other bits are reserved.

`<mask>` The decimal weight of the enabled bits.

Query `UEE?`

The query returns the current decimal value in the User Event Enable register.

Returned Format `[UEE] <mask><NL>`

UER?

Query `UER?`

This query returns the current value of the User Event register as a decimal number. Bit 0 (LCL - Remote/Local change) is used. All other bits are reserved.

Returned Format `[UER?] <value><NL>`

VIEW

Command

```
:VIEW {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | PMEMory<number> | HISTogram |
FFT}
```

The VIEW command turns on a channel, function, pixel memory, waveform memory, histogram, or FFT.

`<number>` For channels: 1, 2, 3, or 4.
For functions: 1 or 2.
For waveform memories (WMEMory): 1, 2, 3, or 4.
For pixel memories (PMEMory): 1.

Example

The following example turns on channel 1.

```
10  OUTPUT 707;":VIEW CHANNEL1"
20  END
```

See Also

The BLANk command turns off a channel, function, pixel memory, waveform memory, or FFT.

10

System Commands

System Commands

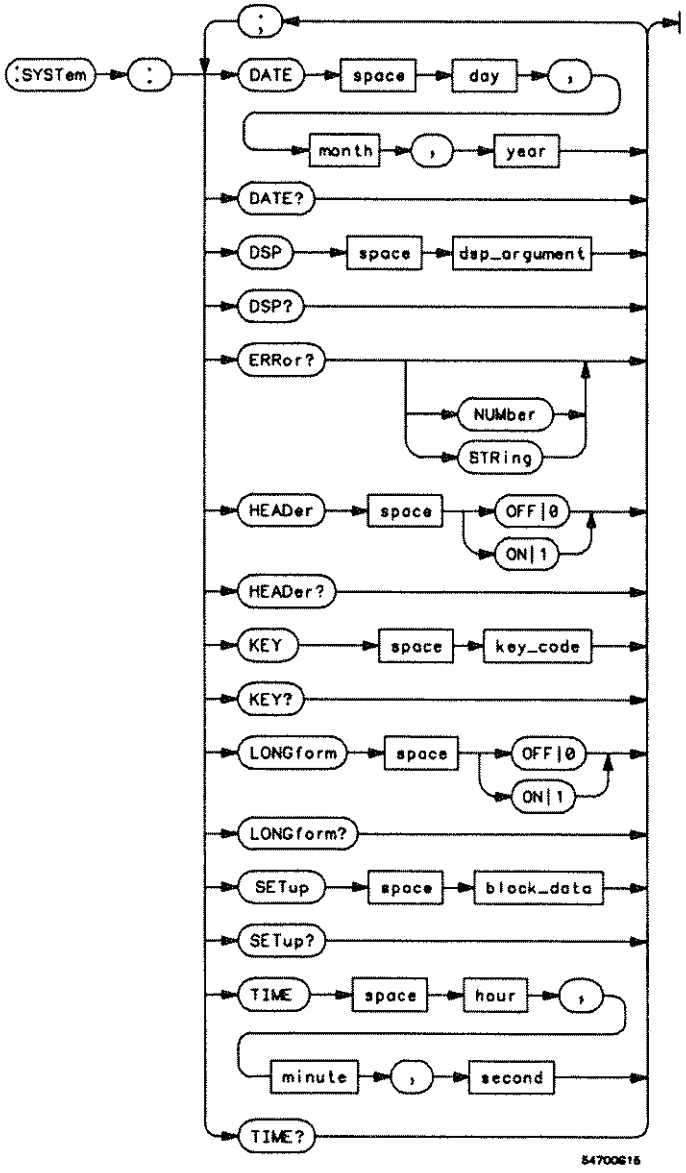
System subsystem commands control the way in which query responses are formatted, simulate front-panel key presses, send and receive setup strings, and enable reading and writing to the advisory line of the oscilloscope. The date and time in the oscilloscope can also be set and read using the System subsystem commands.

The System subsystem contains the following commands and queries:

- DATE,
- DSP,
- ERRor?,
- HEADer,
- KEY,
- LONGform,
- SETup, and
- TIME.

Figure 10-1 is the syntax diagram for the System subsystem commands.

Figure 10-1



System Commands Syntax Diagram

DATE

Command

```
:SYSTem:DATE <day>,<month>,<year>
```

The SYSTem:DATE command sets the date in the oscilloscope and is not affected by the *RST common command.

`<year>` <yyyy> | <yy>

`<month>` <1, 2, . . . 12> | <JAN, FEB, MAR . . .>

`<day>` <1 . . . 31>

Example

The following example sets the date to February 1, 1993.

```
10  OUTPUT 707;":SYSTEM:DATE 1,2,93"
20  END
```

Query

```
:SYSTem:DATE?
```

The SYSTem:DATE query returns the current date in the oscilloscope.

Returned Format

```
[:SYSTem:DATE] <day> <month> <year>>
```

Example

The following example queries the date.

```
10 DIM DATE$ [50]
20 OUTPUT 707;":SYSTEM:DATE?"
30 ENTER 707; DATE$
40 PRINT DATE$
```

DSP

Command

```
:SYSTem:DSP <string>
```

The SYSTem:DSP command writes a quoted string, excluding quotes, to the advisory line of the instrument display. If you want to clear a message on the advisory line, send a null (empty) string.

`<string>` An alphanumeric character array up to 64 bytes long.

Example

The following example writes the message, "Test 1," to the advisory line of the oscilloscope.

```
10  OUTPUT 707;":SYSTEM:DSP ""Test 1"""
20  END
```

Query

```
:SYSTem:DSP?
```

The SYSTem:DSP query returns the last string written to the advisory line. This may be a string written with a SYSTem:DSP command or an internally generated advisory.

The string is actually read from the message queue. The message queue is cleared when it is read. Therefore, the displayed message can only be read once over the bus.

Returned Format

```
[:SYSTem:DSP] <string><NL>
```

Example

The following example places the last string written to the advisory line of the oscilloscope in the string variable, Advisory$. Then, it prints the contents of the variable to the controller's screen.

```
10  DIM Advisory$[64]          !Dimension variable
20  OUTPUT 707;":SYSTEM:DSP?"
30  ENTER 707;Advisory$
40  PRINT Advisory$
50  END
```

ERRor?

Query `:SYSTem:ERRor? [{NUMBer | STRing}]`

The SYSTem:ERRor query outputs the next error number in the error queue over the HP-IB. When either NUMBer or no parameter is specified in the query, only the numeric error code is output. When STRing is specified, the error number is output followed by a comma and a quoted string describing the error. Table 10-1 lists the error numbers and their corresponding error messages. The error messages are also listed in the "Error Messages" chapter where possible causes are given for each message.

Returned Format `[:SYSTem:ERRor] <error_number>[,<quoted_string>]<NL>`

`<error_number>` A numeric error code.

`<quoted_string>` A quoted string describing the error.

Example The following example reads the oldest error number and message in the error queue into the string variable, Condition$, then prints the contents of the variable to the controller's screen.

```
10  DIM Condition$[64]        !Dimension variable
20  OUTPUT 707;":SYSTEM:ERROR? STRING"
30  ENTER 707;Condition$
40  PRINT Condition$
50  END
```

This oscilloscope has an error queue that is 30 errors deep and operates on a first-in, first-out (FIFO)basis. Successively sending the SYSTem:ERRor query returns the error numbers in the order that they occurred until the queue is empty. When the queue is empty, this query returns headers of 0, "No error." Any further queries return zeros until another error occurs. Note that front-panel generated errors are also inserted in the error queue and the Event Status Register.

Send the *CLS common command to clear the error queue and Event Status Register before you send any other commands or queries.

See Also

"Error Messages" chapter for more information on error messages and their possible causes.

Table 10-1 **Error Messages**

Error Number	Description	Error Number	Description
0	No error	−158	String data not allowed
−100	Command error	−160	Block data error
−101	Invalid character	−161	Invalid block data
−102	Syntax error	−168	Block data not allowed
−103	Invalid separator	−170	Expression error
−104	Data type error	−171	Invalid expression
−105	GET not allowed	−178	Expression data not allowed
−108	Parameter not allowed	−200	Execution error
−109	Missing parameter	−222	Data out of range
−112	Program mnemonic too long	−223	Too much data
−113	Undefined header	−310	System error
−121	Invalid character in number	−350	Too many errors
−123	Numeric overflow	−400	Query error
−124	Too many digits	−410	Query INTERRUPTED
−128	Numeric data not allowed	−420	Query UNTERMINATED
−131	Invalid suffix	−430	Query DEADLOCKED
−138	Suffix not allowed	−440	Query UNTERMINATED after indefinite response
−141	Invalid character data		
−144	Character data too long		

HEADer

Command

```
:SYSTem:HEADer {{ON | 1} | {OFF | 0}}
```

The SYSTem:HEADer command specifies whether the instrument will output a header for query responses. When SYSTem:HEADer is set to ON, the query responses include the command header.

Example

The following example sets up the oscilloscope to output command headers with query responses.

```
10  OUTPUT 707;":SYSTEM:HEADER ON"
20  END
```

Query

```
:SYSTem:HEADer?
```

The SYSTem:HEADer query returns the state of the SYSTem:HEADer command.

Returned Format

```
[:SYSTem:HEADer] {1 | 0}<NL>
```

Example

The following example checks the current status of headers and places the result in the string variable "Result$." Then it prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]           !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

Turn headers off when returning values to numeric variables. Headers are always off on all COMMON command queries because headers are not defined in the IEEE 488.2 standard.

KEY

Command

```
:SYSTem:KEY <key_code>
```

The SYSTem:KEY command simulates the pressing of a specified front-panel key. SYSTem:KEY commands may be sent over HP-IB in any order that are legal key presses from the front panel. Make sure the instrument is in the desired state before executing the SYSTem:KEY command.

`<key_code>` An integer, as described in table 10-2.

> The positions of softkeys will change in future firmware releases. Programs that depend on the order of softkey presses may not be compatible with future firmware releases.

CAUTION Use of the :SYSTem:KEY commands can place the instrument into undefined states. We recommend using the specific command for the action desired. The definition of keys varies with each menu and could change with a new firmware revision. Programs using these commands might not be transportable.

Example The following example performs an AUTOSCALE operation by simulating the front-panel AUTOSCALE key.

```
10  OUTPUT 707;":SYSTEM:KEY 2"
20  END
```

Query

```
:SYSTem:KEY?
```

The SYSTem:KEY query returns the key code for the last key pressed on the front panel.

Returned Format

```
[:SYSTem:KEY] <key_code>
```

> The shift key is not accessible over the HP-IB. Shifted keys have their own codes as listed in table 10-2.

Table 10-2 **Front-Panel Key Codes**

Key	Key Code
CLEAR DISPLAY	58
RUN	50
STOP SINGLE	42
DISK	34
WAVEFORM	26
SETUP	18
PRINT	10
AUTOSCALE	2
Arrow Keys	
RIGHT ARROW	5
LEFT ARROW	13
Numeric Keypad	
0	43
1	35
2	60
3	20
4	27
5	3
6	12
7	19
8	11
9	4

Key	Key Code
Numeric entry controls	
ENTER	21
MILLI	29
MICRO	37
NANO	45
PICO	53
POINT	52
SIGN	28
CLEAR	44
EEX	36
Menu Select Keys	
TIMEBASE	56
TRIGGER	48
ACQUISITION	40
DISPLAY	32
MARKER	24
DEFINE MEASURE	16
MATH	8
APPLICATION	0
HELP	57
UTILITY	49

Key	Key Code
Softkey key codes	
Softkey 0 (top key)	59
Softkey 1	1
Softkey 2	9
Softkey 3	17
Softkey 4	25
Softkey 5	33
Softkey 6 (bottom key)	41
Plug-in keys	
1A	129
2A	130
3A	131
4A	132
1B	133
2B	134
3B	135
4B	136
1C	137
2C	138
3C	139
4C	140
1D	141
2D	142
3D	143
4D	144

Key	Key Code
Shifted Keys (64 + unshifted key value)	
FINE	77
SETUP_PRINT	74
LOCAL	106
UNDO_AUTOSCALE	66
RISETIME	83
FALLTIME	75
FFT	72
FREQUENCY	68
LIMIT TEST	80
MASK TEST	88
HISTOGRAM	96
PERIOD	85
+ WIDTH	91
– WIDTH	67
DUTY CYCLE	76
DELTA TIME	93
V P-P	99
V MIN	124
V MAX	84
V RMS	101
V AMPTD	107
V BASE	116
V TOP	92
MORE MEAS	109

Key	Key Code
CLEAR MEASURE	108
PRESHOOT	100
OVERSHOOT	117

LONGform

Command

```
:SYSTem:LONGform {{ON | 1} | {OFF | 0}}
```

The SYSTem:LONGform command specifies the format for query responses. If the LONGFORM is set to OFF, command headers and alpha arguments are sent from the oscilloscope in the short form (abbreviated spelling). If LONGFORM is set to ON, the whole word is output.

Example

The following example sets the format for query responses from the oscilloscope to the short form (abbreviated spelling).

```
10  OUTPUT 707;":SYSTEM:LONGFORM OFF"
20  END
```

Query

```
:SYSTem:LONGform?
```

The SYSTem:LONGform query returns the current state of the SYSTem:LONGform command.

Returned Format

```
[:SYSTem:LONGform] {0 | 1}<NL>
```

Example

The following example checks the current format for query responses from the oscillscope and places the result in the string variable, Result$, then it prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]                  !Dimension variable
20  OUTPUT 707;":SYSTEM:LONGFORM?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

This command has no effect on input headers and arguments sent to the instrument. Headers and arguments may be sent to the instrument in either the long form or short form regardless of the current state of the LONGFORM command.

SETup

Command

```
:SYSTem:SETup <binary block data>
```

The SYSTem:SETup command sets up the instrument as defined by the data in the setup string from the controller.

Example

The following example sets up the instrument as defined by the setup string stored in the variable, Set$.

```
10  OUTPUT 707 USING "#,-K";":SYSTEM:SETUP ";Set$
20  END
```

> # is an HP BASIC image specifier that suppresses the automatic output of the EOI sequence following the last output item.
>
> K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trailing blanks.

Query

```
:SYSTem:SETup?
```

The SYSTem:SETup query outputs the instrument's current setup to the controller in binary block data format as defined in the IEEE 488.2 standard.

Returned Format

```
[:SYSTem:SETup] #NX...X<setup data string>
```

The first character in the setup data string is a magic number added for disk operations.

Example

The following example stores the current instrument setup in the string variable, Set$.

```
10  DIM Set$[15000]                    !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER OFF"  !Response headers off
30  OUTPUT 707;":SYSTEM:SETUP?"
40  ENTER 707 USING "-K";Set$
50  END
```

> –K is an HP BASIC image specifier which places the block data in a string, including carriage returns and line feeds, until EOI is true or when the dimensioned length of the string is reached.

See Also

When headers and LONGform are on, the SYSTem:SETup query operates the same as the *LRN query in the common commands. Otherwise, *LRN and SETup are not interchangeable.

TIME

Command

```
:SYSTem:TIME <hour>,<minute>,<second>
```

The SYSTem:TIME command sets the time in the instrument and is not affected by the *RST common command.

Example

```
10  OUTPUT 707;":SYSTEM:TIME 10,30,45"
20  END
```

Query

```
:SYSTem:TIME?
```

The SYSTem:TIME query returns the current time in the instrument.

Returned Format

```
[:SYSTem:TIME] <hour>,<minute>,<second>
```

11

Acquire Commands

Acquire Commands

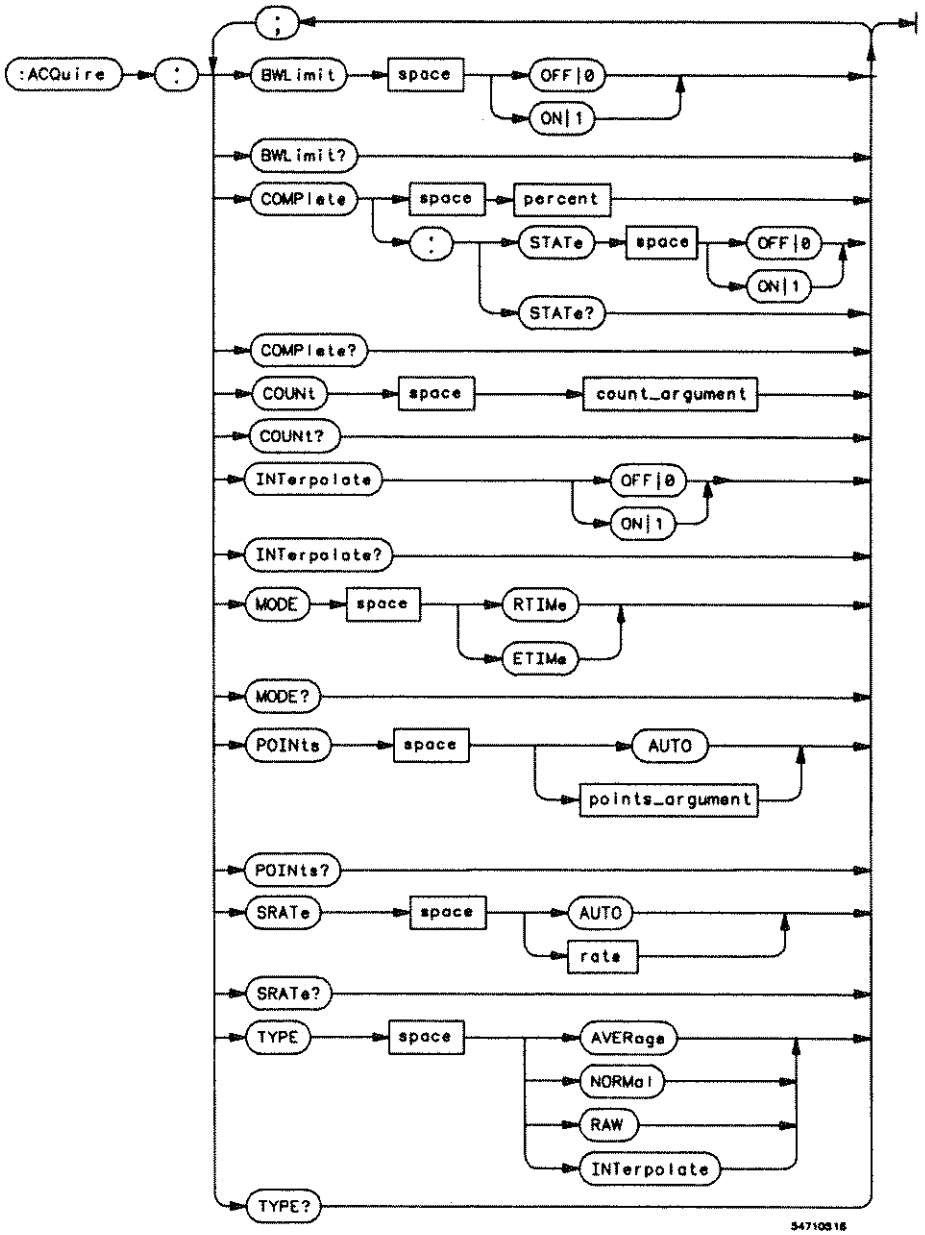
The ACQUIRE subsystem commands set up conditions for executing a DIGITIZE root level command to acquire waveform data. The commands in this subsystem select the type of data, the number of averages, and the number of data points.

The Acquire Subsystem contains the following commands and queries:

- BWLimit,
- COMPlete,
- COMPlete:STATe,
- COUNt,
- INTerpolate,
- MODE,
- POINts (record length),
- SRATe (SampleRATE), and
- TYPE.

Figure 11-1 is the syntax diagram for the Acquire subsystem commands.

Figure 11-1



Acquire Subsystem Commands Syntax Diagram

percent	an integer, 0 to 100
count_argument	an integer, 1 to 4096
points_argument	an integer, (see the POINts command for valid ranges)
rate	a floating point sample rate

Acquire Subsystem Commands Syntax Diagram (continued)

BWLimit

Command

```
:ACQuire:BWLimit {{ON | 1 } | {OFF | 0 }}
```

The :ACQuire:BWLimit command controls the selection of filtering. The bandwidth limit filter only applies to data acquired in the real-time acquisition mode. This command does not pertain to the equivalent-time mode.

ON The digital bandwidth limit filter passes the raw data through a filter which limits the bandwidth to approximately $\frac{f_s}{20}$, where f_s is the current sample frequency.

OFF The filter is turned off.

Example

The following example turns the bandwidth limit filter off.

```
10  OUTPUT 707;":ACQUIRE:BWLIMIT OFF"
20  END
```

Query

```
:ACQuire:BWLimit?
```

The ACQuire:BWLimit query returns the current bandwidth limiting state.

Returned Format

```
[:ACQuire:BWLimit] {ON | OFF}<NL>
```

Example

The following example places the current setting of the bandwidth limit filter in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":ACQUIRE:BWLIMIT?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

COMPlete

Command

```
:ACQuire:COMPlete <percent>
```

The ACQuire:COMPlete command specifies how many of the data point storage bins (time buckets) in the waveform record must contain a waveform sample before a measurement will be made. For example, if ACQuire:COMPlete ON has been sent, and then :ACQuire:COMPlete 60, then 60% of the storage bins in the waveform record need to contain a waveform data sample.

If the ACQUIRE:TYPE is set to NORMal, RAW, or INTerpolate, the oscilloscope only needs one value per time bucket for that time bucket to be considered full.

If the ACQUIRE:TYPE is set to AVERage, each time bucket must have n hits for it to be considered full, where n is the value set by ACQUIRE:COUNT.

Due to the nature of real time acquisition, 100% of the waveform record bins are filled after each trigger event and all of the previous data in the record is replaced by new data. Hence, the complete mode really has no effect and the behavior of the oscilloscope is the same as when the completion criteria is set to 100%.

The range of the COMPLETE command is 0 to 100, and indicates the percentage of time buckets that must be full before the acquisition is considered complete. If the complete value is set to 100%, all time buckets must contain data for the acquisition to be considered complete. If the complete value is set to 0, then one acquisition cycle will take place. Completion defaults to 90.

`<percent>` An integer, 0 to 100, representing the percentage of storage bins (time buckets) that must be full before an acquisition is considered complete.

Example

The following example sets the completion criteria for the next acquisition to 90%.

```
10  OUTPUT 707;":ACQUIRE:COMPLETE 90"
20  END
```

Query

```
:ACQuire:COMPlete?
```

The ACQuire:COMPlete query returns the completion criteria for the currently selected TYPE.

Returned Format

```
[:ACQuire:COMPlete] <percent>
```

`<percent>` An integer, 0 to 100, representing the percentage of time buckets that must be full before an acquisition is considered complete.

Example

The following example reads the completion criteria and places the result in the string variable, Percent$. Then, it prints the content of the variable to the controller's screen.

```
10  DIM Percent$[50]                    !Dimension variable
20  OUTPUT 707;":ACQUIRE:COMPLETE?"
30  ENTER 707;Percent$
40  PRINT Percent$
50  END
```

COMPlete:STATe

Command

```
:ACQuire:COMPlete:STATe {{ON  |1}{OFF |0}}
```

This command specifies the state of the :ACQuire:COMplete mode. This mode is used to make a trade-off between how often equivalent time waveforms are measured and how much new data is included in the waveform record when a measurement is made. This command has no effect when the oscilloscope is in real time mode since the entire record is filled on every trigger. However, in equivalent time mode as few as 0 new data points will placed in the waveform record as the result of any given trigger event. The acquire mode of the oscilloscope is set with the ACQuire:MODE command.

> This command is used only when the oscilloscope is operating in equivalent time mode and a digitize is not being performed. The DIGitize command temporarily overrides the setting of this mode and forces it to ON.

ON Turns the complete mode on and completion percent can be specified.

OFF When off, the oscilloscope makes measurements on waveforms after each acquisition cycle, regardless of how complete they are. The waveform record is not cleared after each measurement, rather previous data points will be replaced by new samples as they are acquired.

Query

```
:ACQuire:COMPlete:STATe?
```

The query returns the state of the ACQuire:COMPlete mode.

COUNt

Command

```
:ACQuire:COUNt <value>
```

The ACQuire:COUNt command sets the ensemble count for ensemble type waveforms. In the AVERage mode, the ACQuire:COUNt command specifies the number of data values to be averaged for each time bucket before the acquisition is considered complete for that time bucket. In the NORMal, RAW, and INTerpolate acquisition types, the ensemble count is ignored and always returns 1.

<value> An integer, 1 to 4096, specifying the number of data values to be averaged.

Example

The following example specifies that 16 data values must be averaged for each time bucket to be considered complete. The number of time buckets that must be complete for the acquisition to be considered complete is specified by the :ACQuire:COMPlete command.

```
10  OUTPUT 707;":ACQUIRE:COUNT 16"
20  END
```

Query

```
:ACQuire:COUNt?
```

The ACQuire:COUNt query returns the currently selected count value.

Returned Format

```
[:ACQuire:COUNt] <value><NL>
```

<value> An integer, 1 to 4096, specifying the number of data values to be averaged.

Example

The following example checks the currently selected count value and places that value in the string variable, Result$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Result$[50]                 !Dimension variable
20  OUTPUT 707;":ACQUIRE:COUNT?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

INTerpolate

Command

```
:ACQuire:INTerpolate {{ON|1}{OFF | 0}}
```

The INTerpolate command turns the interpolator on or off when the oscilloscope is in real time acquisition mode. This command performs the same function as the ACQuire:TYPE INTerpolate command and is included for compatability with previous instruments.

Query

```
:ACQuire:INTerpolate?
```

The query returns the current state of the interploate control.

Returned Format

```
[:ACQuire:INTerpolate] {1 | 0}<NL>
```

MODE

Command

```
:ACQuire:MODE {RTIMe | ETIMe}
```

The ACQuire:MODE command sets the acquisition mode of the oscilloscope. Sampling mode can be Equivalent TIMe (Repetitive) or Real TIMe. In the Real TIMe mode, the complete data record is acquired on a single trigger event. In the Equivalent TIMe mode, the data record is acquired over multiple trigger events.

Example

The following example sets the acquisition mode to real time.

```
10  OUTPUT 707;":ACQUIRE:MODE RTIME"
20  END
```

Query

```
:ACQuire:MODE?
```

The ACQuire:MODE query returns the current acquisition sampling mode.

Returned Format

```
[:ACQuire:MODE] {RTIMe | ETIMe}<NL>
```

Example

The following example places the current acquisition mode in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]              !Dimension variable
20  OUTPUT 707;":ACQUIRE:MODE?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

POINts

Command

```
:ACQuire:POINts {AUTO | <points_value>}
```

The ACQuire:POINts command specifies the requested record length for an acquisition. The actual number of points in an acquisition record may be fewer than requested. Therefore, always query the points value with the WAVeform:POINts query or WAVeform:PREamble to determine the actual number of acquired points.

The points value can be set to AUTO which allows the oscilloscope to select the number of points based upon the sample rate and time base scale.

<points_value> An integer representing the record length.

In real-time mode, the range of points available for a single channel depends on the oscilloscope model and plug-in configuration, as shown in the following table:

Table 11-1 Points Value Ranges

	HP 54710A/HP 54720A	HP 54710D/HP 54720D
Single-wide plug-in	16 to 16384	16 to 65536
Double-wide plug-in	16 to 32768	16 to 131072
Four-wide plug-in	16 to 65536	16 to 262144

Note: the HP 54722 four-wide plug-in cannot be used in HP 54710A/HP 54710D mainframes.

In equivalent-time mode, the points value range is 16 to 32768 points for any model.

Example

The following example sets the record length to 500 points.

```
10  OUTPUT 707;":ACQUIRE:POINTS 500"
20  END
```

Query `:ACQuire:POINts?`

The ACQuire:POINts query returns the requested record length.

Returned Format `[:ACQuire:POINts] <points_value><NL>`

Example

The following example checks the current setting for record length and places the result in the string variable, Length$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Length$[50]                  !Dimension variable
20  OUTPUT 707;":ACQUIRE:POINTS?"
30  ENTER 707;Length$
40  PRINT Length$
50  END
```

See Also :WAVeform:DATA command

SRATe (Sample RATe)

Command

```
:ACQuire:SRATe {AUTO | <rate>}
```

The ACQuire:SRATe command sets the acquisition sample rate of the oscilloscope. It is always set to AUTO when the oscilloscope is in the equivalent time mode.

`AUTO` The AUTO rate allows the oscilloscope to select a sample rate that best accommodates the selected record length and sweep speed.

`<rate>` A real number representing the sample rate. Any value can be sent, but the value is rounded to the next fastest sample rate available for the given plug-in.

Table 11-2 **Sample Rates**

Single-wide plug-in	0.5 to 2.0E+9 (floating-point, NR3 format)
Double-wide plug-in	0.5 to 4.0E+9 (floating-point, NR3 format)
Four-wide plug-in	0.5 to 8.0E+9 (floating-point, NR3 format)

Table 11-3

Sample Rates

.5	1	2.5	5	10	25	50	100	250	500	1K	2.5K	5K	10K	25K
50K	1M	2.5M	5M	10M	20M	50M	100M	250M	500M	1G	2G	4G	8G	

Example

The following example sets the sample rate to 250 MHz.

```
10  OUTPUT 707;":ACQUIRE:SRATE 250E+6"
20  END
```

Query

```
:ACQuire:SRATe?
```

The ACQuire:SRATe query returns the current acquisition sample rate.

Returned Format

```
[:ACQuire:SRATe] {AUTO | <rate>}<NL>
```

Example

The following example places the current sample rate in the string variable, Sample$, then prints the contents of the variable to the controller's screen.

```
10  DIM Sample$[50]           !Dimension variable
20  OUTPUT 707;":ACQUIRE:SRATE?"
30  ENTER 707;Sample$
40  PRINT Sample$
50  END
```

TYPE

Command

```
:ACQuire:TYPE {NORMal | RAW | INTerpolate |
AVERage}
```

The ACQuire:TYPE command, in conjunction with the acquisition mode, selects the type of acquisition that takes place when a DIGITIZE root level command is executed.

RAW The RAW acquisition type applies only to the real time acquisition mode. In this acquisition type, one data point for each time bucket is acquired. No interpolation occurs and only the data points acquired are in the record. If this type is requested when the instrument is in equivalent time mode, this type will be active when the instrument is placed in real time mode.

INTerpolate The INTERPOLATE acquisition type is only available when the acquisition mode is set to real time. With this acquisition type, one data point for each time bucket is stored and additional data points are filled in between the acquired data points by interpolation.

NORMal The NORMAL acquisition type can only be selected if the instrument is in equivalent time mode. With this acquisition type, the last point in each bucket is stored. No averaging or interpolation is performed for this type.

AVERage The AVERAGE acquisition type can only be selected if the instrument is in equivalent time mode. With this acquisition type, the data consists of the average of the last n hits in a time bucket, where n is the current count value as set by the ACQuire:COUNt command.

Example

The following example sets the acquisition type to NORMAL.

```
10  OUTPUT 707;":ACQUIRE:TYPE NORMAL"
20  END
```

Query

```
:ACQuire:TYPE?
```

The ACQuire:TYPE query returns the current acquisition type.

Returned Format

```
[:ACQuire:TYPE] {NORMal | RAW | INTerpolate | AVERage}<NL>
```

Example

The following example places the current acquisition type in the string variable, Type$, then prints the contents of the variable to the controller's screen.

```
10  DIM Type$[50]                          !Dimension variable
20  OUTPUT 707;":ACQUIRE:TYPE?"
30  ENTER 707;Type$
40  PRINT Type$
50  END
```

12

Calibration Commands

Calibration Commands

This section briefly explains the calibration of the HP 54720 and HP 54710 oscilloscopes. It is intended to give you or the calibration lab personnel an understanding of the various calibration levels available, and how they were intended to be used. Also, this section acquaints you with the terms used in this manual, the help screens, and the datasheets.

Mainframe Calibration

Mainframe calibration allows the oscilloscope to establish the calibration factors for each slot independent of the plug-ins. These factors are stored in the oscilloscope's nonvolatile RAM. Mainframe calibration is initiated from the "Utility/Calibrate/Calibrate Mainframe" menu, or from the key on the front panel of the HP 54717A Calibration Plug-in.

Mainframe calibration should be done periodically (at least annually), or if the ambient temperature since the last mainframe calibration has changed more than ±5 °C. The temperature change since the last calibration is shown on the calibration status screen which is found under the "Utility/Calibrate/Calibrate Status On" menu. It is the line labeled "Current Frame Temperature Δ: _ °C."

Th equipment needed to perform mainframe calibration is:

- HP 54717A Calibration Plug-in
- Three BNC cables (1 to 2 ft, two cables must be the same length)
- 20-dB BNC attenuator
- BNC Tee

When the calibration is initiated, instructions appear on the screen on how to connect the various pieces of equipment to perform the calibration. Note: you can use the 20-dB attenuator as a 50-Ω terminator.

There is a switch on the back panel of the oscilloscope that allows the mainframe calibration to be enabled or disabled. The rear panel has a drawing that shows the function of each switch and which is the protected position. To prevent access to the mainframe calibration switch, place a sticker over the access hole to this switch.

See Also

The Service Guide has more details about the mainframe calibration and the position of the rear-panel memory protect switches.

Plug-in Calibration

Plug-in calibration allows the oscilloscope to establish the calibration factors for a particular plug-in independent of the mainframe in which it is calibrated. These calibration factors are stored in the plug-in's EEPROM, so the factors stay with the plug-in, not with the mainframe in which the plug-in was calibrated. Plug-in calibration is initiated from the "Utility/Calibrate/ Calibrate Plug-in" menu.

Plug-in calibrations should be done annually or if the plug-in has been repaired or reprogrammed.

Because the intent is for the oscilloscope to determine the calibration factors for the plug-in only, plug-in calibrations should be done right after a mainframe calibration has been done so any drifts in the mainframe calibration (due to time or temperature) do not cause drift in the calibration factors assigned to the plug-in.

The equipment needed to perform a plug-in calibration is a BNC cable (1 to 2 ft).

Inside the plug-in, there is a switch that enables or disables the ability to write into the plug-in's EEPROM memory. You must remove the cover of the plug-in to gain access to the switch. To prevent access to the switch, place a sticker on the plug-in cover.

Normal Accuracy Calibration Level

Normal accuracy calibration level refers to the level of accuracy that is achieved when a plug-in is installed in a slot of a mainframe, but has not been calibrated to best accuracy in that particular slot.

The oscilloscope uses the mainframe calibration factors and the plug-in calibration factors to determine the normal calibration factors for that plug-in in that slot.

The intent of this level of calibration is to allow any plug-in to be inserted in any mainframe slot and still achieve a typical vertical accuracy of ±3%.

For two-wide plug-ins, such as the HP 54721A, and four-wide plug-ins, the interleaving of the digitizers in the mainframe slots is performed during the best-accuracy calibration. Therefore, the real-time sampling mode is typically not useful in the normal accuracy mode. For accurate results, a best accuracy calibration must be performed to use two-wide or four-wide plug-ins in the real-time mode.

See also

"Best Accuracy" in this chapter gives details on the best-accuracy calibration.

Best Accuracy Calibration Level

Best accuracy calibration level refers to the level of accuracy that is achieved when a plug-in is installed in a mainframe slot and is calibrated to the best-accuracy level. A best-accuracy calibration is initiated from the "Channel/Calibrate/Calibrate to best accuracy" menu.

The intention is that a recent best-accuracy calibration should be performed before critical measurements are made to ensure the best measurement results. The best-accuracy calibration is fairly quick and easy, and only a BNC cable is needed to perform the calibration.

The oscilloscope establishes best-accuracy calibration factors by calibrating the plug-in and the mainframe slot as a system. These calibration factors are stored in the mainframe's nonvolatile RAM. The oscilloscope keeps these calibration factors for a plug-in/slot combination until a plug-in with a different serial number is calibrated to best accuracy in that slot. This means that a plug-in that was calibrated to best accuracy in a particular mainframe slot can be removed and reinstalled later in that slot and the best accuracy calibration is still in effect, as long as no other plug-in was placed in that slot and calibrated to the best accuracy level in the meantime.

The best accuracy calibration factors for a multi-slot plug-in are stored in the left slot. For example, if you performed a best accuracy calibration on a two-wide plug-in in slots 2 and 3, the best accuracy calibration factors are stored in the memory behind slot two. You can remove the two-wide plug-in and install a single-wide plug-in in slot 3. You can perform a best accuracy calibration on the single-wide plug-in in slot 3 without affecting the best accuracy calibration factors for the two-wide plug-in that are stored behind slot 2.

For the best measurement results, a channel (plug-in/slot combination) should be calibrated to best accuracy level. Also, two- and four-slot-wide plug-ins must be calibrated to best accuracy level to give accurate results in real-time mode.

The best-accuracy calibration level has a temperature Δ associated with it, just like the mainframe calibration. This temperature Δ appears in the "Plug-in" calibration status screen for all plug-ins under the "Utility/Calibrate/Cal status on" menu and for each plug-in under the "Channel/Calibrate/Cal status on" menu.

In order for a channel to meet the specifications listed under "Best Accuracy" both the "Current frame Temp Δ" and the "Best Accuracy Temp Δ" must be between ±5 °C. In other words, the oscilloscope must be within ±5 °C from the temperature at which the mainframe was calibrated and within ±5 °C from the temperature at which the best-accuracy calibration was performed for a particular plug-in slot calibration.

Probe Calibration

Probe calibration allows the oscilloscope to establish the gain and offset of a probe that is connected to a channel of the oscilloscope, and apply those factors to the calibration of that channel. Probe calibration is initiated from the "Channel/Probe/Calibrate probe" menu. Probe calibration commands are part of the Channel subsystem.

To achieve the specified accuracy (±2%) with a probe connected to a channel, that channel should be calibrated to the best accuracy level. This ensures the best measurement results.

If the best accuracy is not needed, a probe can be calibrated with a channel that is at the normal-accuracy level. This achieves a typical vertical accuracy of ±4%.

For active probes that the oscilloscope can identify through the probe power connector, like the HP 54701A, the oscilloscope automatically adjusts the vertical scale factors for that channel even if a probe calibration is not performed. For passive probes or nonidentified probes, the oscilloscope adjusts the vertical scale factors only if a probe calibration is performed. If you do not perform a probe calibration but want to use a passive probe, the attenuation factor can be entered in the "Channel/Probe/Probe atten" menu.

If the probe being calibrated has an attenuation factor that allows the oscilloscope to adjust the gain (in hardware) to produce even steps in the vertical scale factors, the oscilloscope will do so. If the probe being calibrated has an unusual attenuation, like 3.75, the oscilloscope may have to adjust the vertical scale factors to an unusual number, like 3.75 V/div. Typically, probes have standard attenuation factors such as divide by 10, divide by 20, or divide by 100.

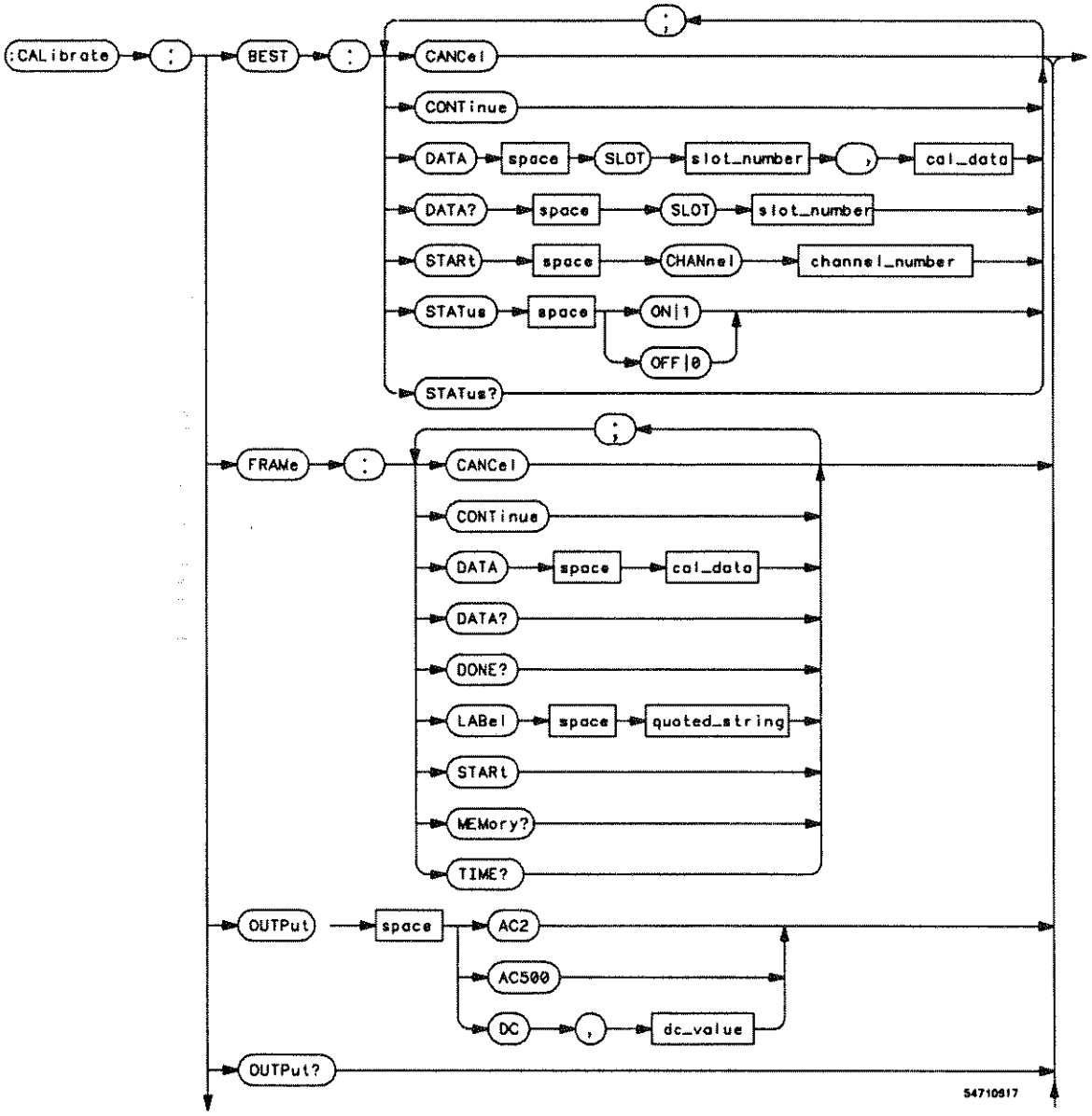
Calibration Commands

The commands in the Calibration subsystem initiate the instrument calibration over the HP-IB. The Calibration Subsystem consists of the following commands and queries:

- BEST:CANcel,
- BEST:CONTinue,
- BEST:DATA,
- BEST:STARt,
- BEST:STATus,
- FRAMe:CANcel,
- FRAMe:CONTinue,
- FRAMe:DATA,
- FRAMe:DONE?,
- FRAMe:LABel,
- FRAMe:STARt,
- FRAMe:MEMory?,
- FRAMe:TIME?,
- OUTPut,
- PLUGin:CANcel,
- PLUGin:CONTinue,
- PLUGin:DONE?,
- PLUGin:MEMory?,
- PLUGin:STARt,
- PLUGin:TIME?
- SKEW, and
- STATus.

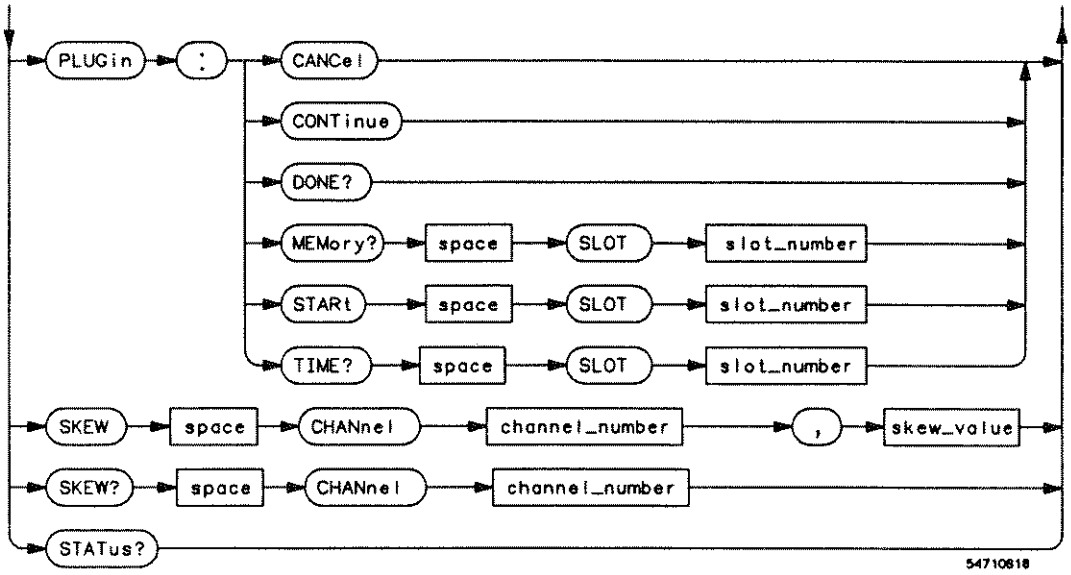
Figure 12-1 is the syntax diagram of the Calibration Subsystem commands.

Figure 12-1



Calibration Subsystem Syntax Diagram

Figure 12-1 (continued)



Calibration Subsystem Syntax Diagram (continued)

BEST:CANcel

Command

```
CALibrate:BEST:CANCel
```

The CALibrate:BEST:CANCel command is equivalent to pressing the Cancel softkey in the front-panel Calibrate best accuracy menu.

BEST:CONTinue

Command

```
CALibrate:BEST:CONTinue
```

The CALibrate:BEST:CONTinue command is equivalent to pressing the front-panel Continue softkey in the Calibrate best accuracy menu.

BEST:DATA

Command

```
CALibrate:BEST:DATA SLOT<slot_number>,
<cal factors>
```

This command sends the BEST data calibration factors to the oscilloscope.

`<slot>` Identifies slot. Allowable values are SLOT1, SLOT2, SLOT3, and SLOT4.

`<cal factors>` Is a <long_block> containing a copy of the BEST data. See "Block Data" in chapter 5 for more information.

Query

```
CALibrate:BEST:DATA? {SLOT<slot_number>}
```

The CALibrate:BEST:DATA? query returns all of the "BEST" accuracy calibration factors associated with the specified channel.

Returned Format

```
[:CALibrate:BEST:DATA]<cal factors>
```

`<cal factors>` A <long_block> containing a copy of the BEST data. See "Block Data" in chapter 5 for more information.

BEST:STARt

Command `CALibrate:BEST:STARt <channel_N>`

The CALibrate:BEST:STARt command starts the best accuracy calibration sequence.

`<channel_N>` Identifies the channel. N specifies a number optionally followed by a letter. If no letter is specified, A is assumed (i.e., CHAN1 = CHAN1A).

BEST:STATus

Command `CALibrate:BEST:STATus {ON | 1 | OFF | 0}`

The CALibrate:BEST:STATus command turns off or on the Best Accuracy Calibration Status display.

Query `:CALibrate:BEST:STATus?`

The query returns the state of the Best Accuracy Calibration Status display.

Returned Format `[:CALibate:BEST:STATus] {1 | 0} <NL>`

FRAMe:CANcel

Command

`CALibrate:FRAMe:CANCel`

The CALibrate:FRAMe:CANCel command is equivalent to pressing the Cancel softkey when in the front-panel Calibrate Frame menu. This command cancels the calibration on the oscilloscope mainframe.

FRAMe:CONTinue

Command

`CALibrate:FRAMe:CONTinue`

The CALibrate:FRAMe:CONTinue command is equivalent to pressing the front-panel CONTINUE softkey when in the frame calibration menu. This command continues the calibration on the oscilloscope mainframe.

FRAMe:DATA

Command `CALibrate:FRAMe:DATA <cal factors>`

This command sends the FRAMe data to the oscilloscope.

`<cal factors>` Is a < long_block> containing a copy of the Frame calibration structure.

Query `CALibrate:FRAMe:DATA?`

The CALibrate:FRAMe:DATA? query returns the calibration factors for the oscilloscope mainframe.

Returned Format `[CALibrate:FRAMe:DATA] <cal factors>`

`<cal factors>` Is a < long_block> containing a copy of the Frame calibration structure.

> You cannot send calibration factors obtained under any version of system software earlier than 3.XX back to an instrument that has been upgraded to system software version 3.XX. The message "Invalid block" will appear and the calibration load will fail. You must recalibrate the instrument under system software version 3.XX to obtain a set of calibration factors that can be loaded using that version of the software.

FRAMe:DONE?

Query `CALibrate:FRAMe:DONE?`

The CALibrate:FRAMe:DONE? query returns the pass/fail status of the last frame calibration.

Returned Format `[:CALibrate:FRAMe:DONE] {1 | 0}`

FRAMe:LABel

Command

```
CALibrate:FRAMe:LABel <label>
```

The CALibrate:FRAMe:LABel command accepts a string of up to 80 characters. It is displayed as part of the frame calibration status screen and is optional. It is intended for user notes, such as name/initials of the calibrator or special notes about the calibration.

`<label>` Is a <quoted string>.

Query

```
:CALibrate:FRAMe:LABel?
```

The query returns the currently defined label for the frame.

Returned Format

```
[CALibrate:FRAMe:LABel]<quoted string><NL>
```

FRAMe:MEMory?

Query

```
CALibrate:FRAMe:MEMory?
```

The CALibrate:FRAMe:MEMory? query returns the state of the frame calibration write protect switch.

Returned Format

```
[:CALibrate:FRAMe:MEMory]    {PROTected|UNPRotected}    <NL>
```

FRAMe:STARt

Command

```
CALibrate:FRAMe:STARt
```

The CALibrate:FRAMe:STARt command starts the annual calibration on the oscilloscope mainframe.

FRAMe:TIME?

Query `CALibrate:FRAMe:TIME?`

The CALibrate:FRAMe:TIME? query returns the date, time and temperature at which the last full frame calibration process was completed.

Returned Format `[:CALibrate:FRAMe:TIME] <time> <NL>`

`<time>` Is in the format DD MMM YY HH:MM <delta_temp>

<delta_temp> is the difference between the current temperature and the temperature when the last calibration was done. For example, <delta_temp> might be:

```
-5C
10C
-12C
```

OUTPut

Command

```
CALibrate:OUTPut {AC2 | AC500} | {DC, <value>}
```

The CALibrate:OUTPut command sets the coupling, frequency and dc level of the calibrator signal output through the front panel CAL connector.

`<value>` Is a dc level value in volts.

Query

```
:CALibrate:OUTPut?
```

The query returns the current setup.

Returned Format

```
[:CALibrate:OUTPut]    {AC2 | AC500} | {DC, <value>}
```

PLUGin:CANcel

Command `CALibrate:PLUGin:CANcel`

The CALibrate:PLUGin:CANcel command is equivalent to pressing the front-panel Cancel softkey when in the Calibrate Plugin menu. This command cancels the calibration on a selected slot (plug-in).

PLUGin:CONTinue

Command `CALibrate:PLUGin:CONTinue`

The CALibrate:PLUGin:CONTinue command is equivalent to pressing the front-panel Continue softkey when in the Calibrate Plugin menu. This command continues the calibration on a selected slot (plug-in).

PLUGin:DONE?

Query `CALibrate:PLUGin:DONE?`

The CALibrate:PLUGin:DONE? query returns the pass/fail status of the last plug-in calibration.

Returned Format `[:CALibrate:PLUGin:DONE] {1 | 0}`

PLUGin:MEMory?

Query `CALibrate:PLUGin:MEMory?{SLOTN}`

The CALibrate:PLUGin:MEMory query returns the state of the plug-in memory write protect switch for the selected slot. The switch must be in the UNPRotected state in order to calibrate the plug-in, or to program the module serial number. If no plug-in is present in the selected slot, the message Empty Slot is returned.

Returned Format `[:CALibrate:PLUGin:MEMory] {PROTected|UNPRotected} <NL>`

PLUGin:STARt

Command `CALibrate:PLUGin:STARt`

The CALibrate:PLUGin:STARt command starts the plug-in calibration sequence.

PLUGin:TIME?

Query `CALibrate:PLUGin:TIME? {SLOTN}`

The CALibrate:PLUGin:TIME? query returns the date, time and temperature at which the plug-in, in slot N, was last calibrated. If no plug-in is present in the selected slot, the message "Empty Slot" is returned.

Returned Format `[:CALibrate:PLUGin:TIME] <value>`

`<value>` Is in the format DD MMM YY HH:MM <delta_temp>

<delta_temp> is the difference between the current temperature and the temperature when the last calibration was done. For example, <delta_temp> might be:

```
-5C
10C
-12C
```

SKEW

Command `CALibrate:SKEW {CHANnelN},<skew>`

The CALibrate:SKEW command sets the channel-to-channel skew factor for channel N. The numerical argument is a real number in seconds which is to be added to the current timebase position to effectively shift the position of CHANnel N's data in time. It can be used to compensate for differences in the electrical lengths of input paths due to cabling and probes.

`<skew>` Is a real number.

Query `:CALibrate:SKEW? {CHANnelN}`

The query returns the current skew factor.

Returned Format `[:CALibrate:SKEW] <skew_factor><NL>`

STATus?

Query `CALibrate:STATus?`

The CALibrate:STATus? query returns the calibration status of the instrument. These are nine, comma-separated, 32-bit integers. The data are available from the front panel via the Display Frame Calibration screen accessed from the Utility calibration menu.

Returned Format `[:CALibrate:STATus] <status>`

`<status>` <Frame Status>,
<Slot1 Vertical>, <Slot 1 Trigger>,
<Slot2 Vertical>, <Slot 2 Trigger>
<Slot3 Vertical>, <Slot3 Trigger>,
<Slot4 Vertical>, <Slot4 Trigger>

13

Channel Commands

Channel Commands

The CHANNEL subsystem commands control all vertical (Y axis) functions of the oscilloscope. The options for the channel subsystem commands vary depending on which plug-in you are using.

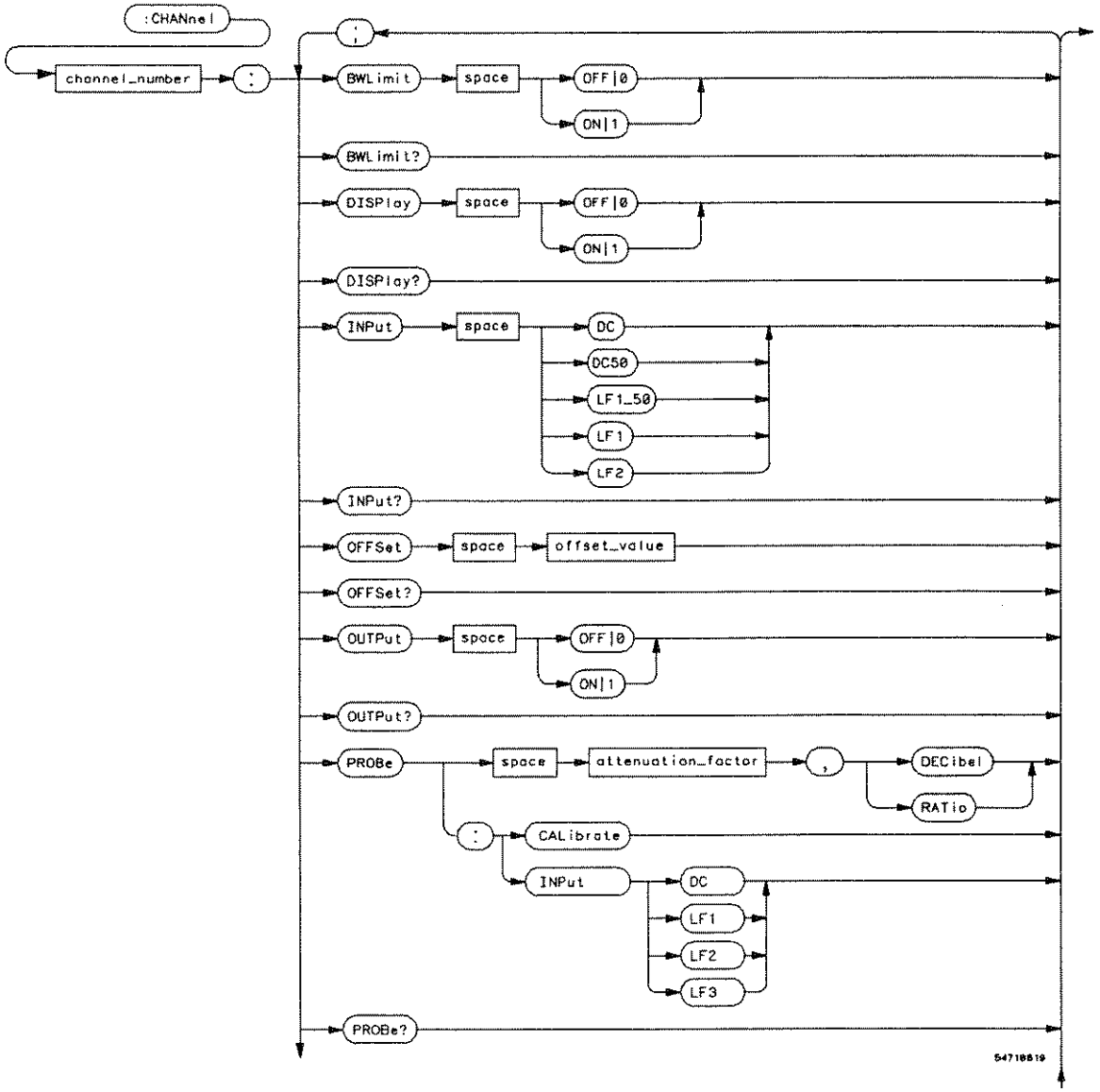
The channel displays may be toggled on and off with the root level commands VIEW and BLANK.

The Channel subsystem contains the following commands and queries:

- BWLimit,
- DISPlay,
- INPut,
- OFFSet,
- OUTPut,
- PROBe,
- PROBe:CALibrate,
- PROBe:INPut,
- PROTection:CLEar,
- PROTection?,
- RANGe,
- SCALe,
- SENSitivity,
- UNITs,
- UNITs:ATTenuation, and
- UNITs:OFFSet.

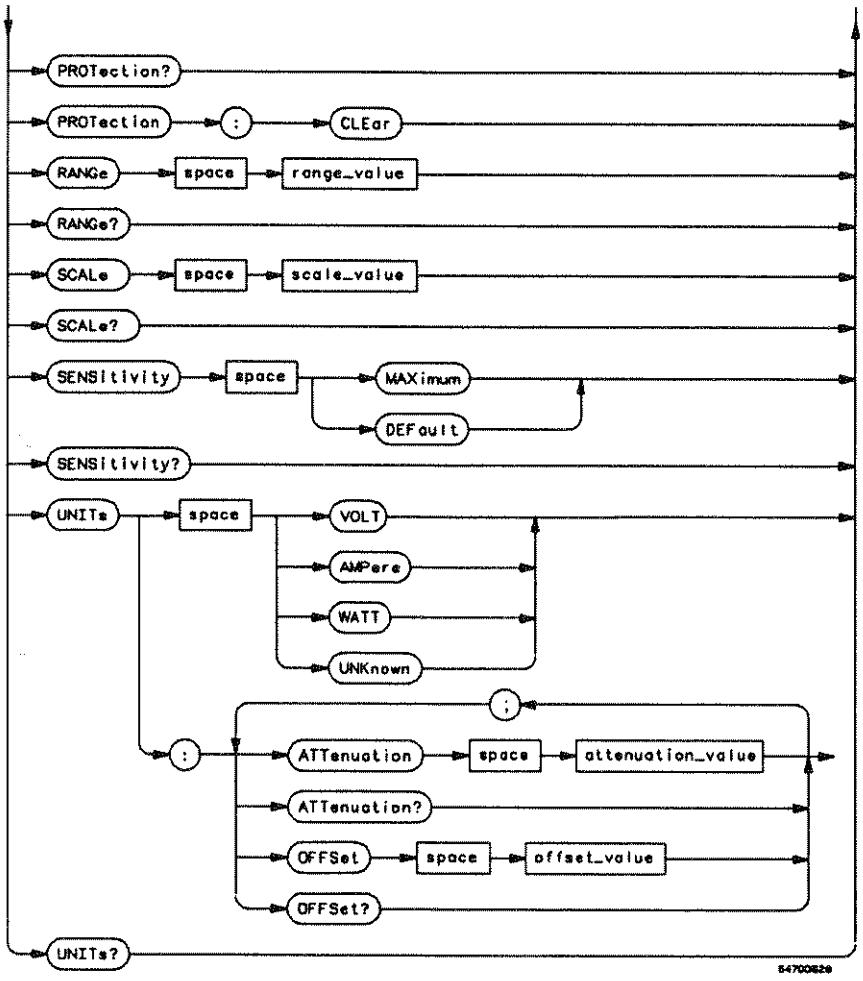
Figure 13-1 is the syntax diagram for the Channel subsystem commands.

Figure 13-1



Channel Subsystem Commands Syntax Diagram

Figure 13-1 (continued)



Channel Subsystem Commands Syntax Diagram (continued)

BWLimit

Command

```
:CHANnel<number> :BWLimit {{ON | 1} | { OFF | 0 }}
```

This command controls an internal low-pass filter if one is present in the channel hardware. When ON, the bandwidth of the specified channel is limited. See the plug-in manual for the cutoff frequency specification. The bandwidth limit filter can be used when either AC or DC coupling has been selected with the :CHANnelN:INPut command.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example sets the internal low-pass filter to "ON" for channel 1.

```
10 OUTPUT 707;":CHANNEL1:BWLimit ON"
20 END
```

Query

```
:CHANnel<number>:BWLimit?
```

The query returns the state of the internal low-pass filter for the specified channel.

Returned Format

```
[:CHANnel<number>:BWLimit] {1 | 0}
```

Example

The following example places the current setting of the internal low pass filter in the string variable, Limit$, and then prints the contents of the variable to the controller's screen.

```
10  DIM Limit$[50]                    !Dimension variable
20  OUTPUT 707;":CHANNEL1:BWLIMIT?"
30  ENTER 707;Limit$
40  PRINT Limit$
50  END
```

DISPlay

Command

```
:CHANnel<number>:DISPlay {{ ON | 1} | { OFF | 0}}
```

This command sets the specified channel to on or off.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example sets channel 1 display to on.

```
10 OUTPUT 707;"CHANNEL1:DISPLAY ON"
20 END
```

Query

```
:CHANnel<number>:DISPlay?
```

The query returns the current display condition for the specified channel.

Returned Format

```
[:CHANnel<number>:DISPlay] {1 | 0}
```

Example

The following example places the current setting of the channel 1 display in the string variable, Display$, and then prints the contents of the variable to the controller's screen.

```
10  DIM Display$[50]                !Dimension variable
20  OUTPUT 707;":CHANNEL1:DISPLAY?"
30  ENTER 707;Display$
40  PRINT Display$
50  END
```

INPut

Command `:CHANnel<number>:INPut <parameter>`

This command selects the input coupling, impedance, and LF/HF reject for the specified channel. The coupling for each channel can be set to AC, DC, or LFReject.

`<number>` The channel number represents an integer 1 - 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<parameter>` The parameters available in this command are dependent on the plug-in and are listed below.

HP 54711A	DC50: dc 50 Ω
HP 54712A	DC50: dc 50 Ω
	LF1_50: ac, 34 kHz
HP 54713A	DC50: dc 50 Ω
	DC: dc 1 MΩ
	LF1: ac 1 MΩ, 90 Hz
	LF2: 1 MΩ, 450 Hz
HP 54713B & HP 54714A	DC50: dc 50 Ω
	DC: dc 1 MΩ
	LF1: ac 1 MΩ, 10 Hz
	LF2: 1 MΩ 450 Hz
HP 54721A	DC50: dc 50 Ω
	LF1_50: ac, >34 kHz
HP 54722A	DC50: dc 50 Ω

Example

The following example sets the channel input to DC50.

```
10 OUTPUT 707;":CHANNEL1:INPut DC50"
20 END
```

Query

```
:CHANnel<number>:INPut?
```

Returned Format

```
[CHANnel<number>:INPut]<parameter>
```

Example

The following example places the current input for channel 1 in the string variable, Input$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Input$[50]            !Dimension variable
20  OUTPUT 707;":CHANNEL1:INPUT?
30  ENTER 707;Input$
40  PRINT Input$
50  END
```

OFFSet

Command

```
:CHANnel<number>:OFFSet <offset value>
```

This command sets the voltage that is represented at center screen for the selected channel. Offset parameters are plug-in, probe, and vertical scale dependent.

`<number>` The channel number represents an integer 1 - 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<offset value>` Offset value at center screen. Usually expressed in volts, but could be in other measurement units such as amperes if you have specified other units using the CHANnel:UNITs command.

Example

The following example sets the offset for a plug-in to 125 m in the current measurement units:

```
10  OUTPUT 707;":CHANNEL1:OFFSET 125E-03"
20  END
```

When you are receiving numeric data into numeric variables, the headers should be turned off. Otherwise, the headers may cause misinterpretation of returned data.

Query

```
:CHANnel<number>:OFFSet?
```

The query returns the current offset value for the specified channel.

Returned Format

[CHANnel<number>:OFFSet] <offset value><NL>

Example

The following example places the offset value of the specified channel in the string variable, Offset$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Offset$[50]
20 OUTPUT 707;"CHANNEL1:OFFSET?"
30 ENTER 707;Offset$
40 PRINT Offset$
50 END
```

OUTPut

Command

```
:CHANnel<number>:OUTPut  {{ON | 1 | OFF 0}}
```

The OUTPut command is used for plug-ins with calibration outputs. It turns the Calibration output on or off.

Query

```
:CHANnel<number>:OUTPut?
```

The query returns the state of the Calibration output.

Returned Format

```
[:CHANnel,number>:OUTPut]  {1 | 0}<NL>
```

PROBe

Command

```
:CHANnel<number>:PROBe <attenuation
factor>{,{RATio | DECibel}]
```

This command sets the probe attenuation factor and, optionally, the units for the probe attenuation factor. The range of the probe attenuation factor is from 0.0001 to 1000000 and −80 dB to 120 dB. The reference constants that are used for scaling the display factors are changed with this command, including automatic measurements and trigger levels.

`<number>` The channel number represents an integer 1 - 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<attenuation factor>` A real number from 0.0001 to 1000000, or − 80 dB to 120 dB representing the probe attenuation factor and depends on the units.

Example

The following example sets the probe attenuation factor of channel 1 to 10 and the units to decibel.

```
10  OUTPUT 707;":CHANNEL1:PROBE 10, DEC"
20  END
```

See Also

For information on skew, see CALibrate commands.

Query

```
:CHANnel<number>:PROBe?
```

The query returns the current probe attenuation setting for the selected channel, and the units.

Returned Format

```
[:CHANnel<number>:PROBe] <attenuation>,{RATio | DECibel}<NL>
```

Example

The following example places the current attenuation setting for channel 1 in the string variable, Atten$, and then the program prints the contents.

```
10  DIM Atten$[50]              !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER OFF"
30  OUTPUT 707;":CHANNEL1:PROBE?
40  ENTER 707;Atten$
50  PRINT Atten$
60  END
```

If you use a string variable, the query returns the attenuation value and the factor (decibel or ratio). If you use an integer variable, the query returns the attenuation value. You must then read the attenuation units into a string variable.

PROBe:CALibrate

Command

```
:CHANnel<number>:PROBe:CALibrate
```

This command starts the probe's calibration for the selected channel.

Example

The following example starts calibration for CHANNEL1.

```
10  OUTPUT 707;":CHANNEL1:PROBE:CALIBRATE"
20  END
```

PROBe:INPut

Command

```
:CHANnel<number>:PROBe:INPut {DC|LF1|LF2|LF3}
```

This command is valid for the HP 54715A plug-in only (with probe attached). It is used to set the input coupling, impedance, and LF reject for the specified channel's probe.

Input impedance is 1MΩ for all settings. DC selects dc coupling. LF1 selects ac coupling with 5 Hz low frequency rejection. LF2 selects ac coupling with 0.5 Hz (500 mHz) low frequency rejection. LF3 selects ac coupling with 0.05 Hz (50 mHz) low frequency rejection.

`<number>` The channel number represents an integer 1 - 4 that specifies the slot in which the channel resides.

Example

The following example sets the probe input to ac coupling at 1MΩ with 5 Hz low frequency rejection.

```
10 OUTPUT 707;":CHANNEL1:PROBe:INPut LF1"
20 END
```

Query

```
:CHANnel<number>:PROBe:INPut?
```

The query returns the current probe input setting of the specified channel.

Returned Format

```
[CHANnelN:PROBe:INPut] {DC|LF1|LF2|LF3} <NL>
```

Example

The following example places the current probe input for channel 1 in the string variable, Probe$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Probe$[50]            !Dimension variable
20  OUTPUT 707;":CHANNEL1:PROBE:INPUT?"
30  ENTER 707;Probe$
40  PRINT Probe$
50  END
```

PROTection:CLEar

Command

```
:CHANnel<number>PROTection:CLEar
```

This command is used to clear (reset) the overload protection. It allows the channel to be used again after the signal that caused the overload has been removed from the channel input.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example

The following example clears the overload protection for channel 1.

```
10  OUTPUT 707;":CHANNEL1:PROTECTION:CLEAR"
20  END
```

PROTection?

Query

```
:CHANnel<number>:PROTection?
```

This query returns the state of the input protection for CHANnel<number>. If a channel has no input protection, then NORMAL is always returned.

Returned Format

```
[CHANnel<number>:PROTection] {NORMal | TRIPped | <null>}
```

Example

The following example places the current state of the input protection for the specified channel in the string variable, Protect$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Protect$[50]
20 OUTPUT 707;"CHANNEL1:PROTECTION?"
30 ENTER 707;Protect$
40 PRINT Protect$
50 END
```

If there is no plug-in in the selected slot, a null string is returned. The message -241, "Hardware missing" will be placed in the error queue.

RANGe

Command

```
:CHANnel<number>:RANGe <range value>
```

The CHANnel<number>:RANGe command defines the full-scale vertical axis of the selected channel. It sets up acquisition and display hardware to display the waveform at a given range scale. The values represent the full scale deflection factor of the vertical axis in volts. These values change as the probe attenuation factor is changed.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<range value>` Full-scale voltage of the specified channel number. Available ranges are dependent on the plug-in being used and attenuation factor.

Example

The following example sets the full-scale range for channel 1 to 500 mV.

```
10  OUTPUT 707;":CHANNEL1:RANGE 500M"
20  END
```

Query

```
:CHANnel<number>:RANGe?
```

The CHANnel<number>:RANGe query returns the current full scale vertical axis setting for the selected channel.

Returned Format

```
[:CHANnel<number>:RANGe]<range value><NL>
```

Example

The following example places the current range value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

SCALe

Command

```
:CHANnel<number>:SCALe <scale value>
```

This command defines the vertical scale of the channel in units per division. This command is the same as the front-panel channel scale.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<scale value>` Vertical scale of the channel in units per division.

Example

The following example sets the scale value for channel 1 to 500 mV.

```
10  OUTPUT 707;":CHANNEL1:SCALE 500M"
20  END
```

Query

```
:CHANnel<number>:SCALe?
```

The query returns the curent scale setting for the specified channel.

Returned Format

```
[:CHANnel<number>:SCALe] <scale value><NL>
```

Example

The following example places the current scale value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:SCALE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

SENSitivity

Command

```
:CHANnel<number>:SENSitivity {MAXimum | DEFault}
```

This command is valid for the HP 54711A plug-in only. It is used to set the sensitivity to either maximum or default values.

<number> The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example The following example sets the sensitivity for channel 1 to maximum.

```
10  OUTPUT 707;":CHANNEL1:SENSITIVITY MAXIMUM"
20  END
```

Query

```
:CHANnel<number>:SENSitivity?
```

The query returns the current setting of the sensitivity of the specified channel.

Returned Format

```
[:CHANnel<number>:SENSitivity] {MAXimum|DEFault}<NL>
```

Example The following example places the current state of the sensitivity for the specified channel in the string variable, Sens$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Sens$[50]
20 OUTPUT 707;"CHANNEL1:SENSITIVITY?"
30 ENTER 707;Sens$
40 PRINT Sens$
50 END
```

UNITs

Command

```
:CHANnel<number>:UNITs {VOLT | AMPere | WATT |
UNKNown}
```

This command allows you to work in vertical units other than volts. The units command changes the name of the Y-axis units from VOLTS to AMPS, WATTs, or UNKNown. The units are implied for other pertinent channel commands (such as RANGe and OFFSet).

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

Example The following example sets the units for channel 1 to amperes.

```
10  OUTPUT 707;":CHANNEL1:UNITS AMPERE"
20  END
```

Query

```
:CHANnel<number>:UNITs?
```

The query returns the current units setting for the specified channel.

Returned Format

```
[:CHANnel<number>:UNITs] {VOLT | AMPere | WATT | UNKNown} <NL>
```

Example The following example places the vertical units for the specified channel in the string variable, Units$, and then prints the contents of the variable to the controller's screen.

```
10 DIM Units$[50]
20 OUTPUT 707;"CHANNEL1:UNITS?"
30 ENTER 707;Units$
40 PRINT Units$
50 END
```

UNITs:ATTenuation

Command

```
:CHANnel<number>:UNITs:ATTenuation <attenuation>
```

This command specifies a ratio that indicates how the unit specified in the :CHANnel <number>:UNITs relates to one volt.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<attenuation>` The number of volts per unit, such as volts/watt.

Example

The following example specifies channel 1 units as watts and that the device under test supplies the channel input with 30 volts for every watt.

```
10 OUTPUT 707;"CHANNEL1:UNITS WATT"
20 OUTPUT 707;"CHANNEL1:UNITS:ATTENUATION 30"
30 END
```

Query

```
:CHANnel<number>:UNITs:ATTenuation?
```

The query returns the number of volts per user specified units.

Returned Format

```
[CHANnel<number>:UNITs:ATTenuation] <attenuation>
```

Example

The following example places the current unit attenuation value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:UNITS:ATTENUATION?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

UNITs:OFFSet

Command

```
:CHANnel<number>:UNITs:OFFSet <offset>
```

This command specifies a constant value in volts that will be added to the signal to compensate for any offset. For example, if the units are watts and the measured signal has 0.5 volts offset at 0 watts, then the argument for this command should be -0.5 to ensure that 0 watts will appear at center screen when CHANnel:OFFSet is set to 0.

`<number>` The channel number represents an integer 1 to 4 followed by an optional letter A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.

`<offset>` The constant value in volts to be added to the measured signal.

Example

The following example sets the offset compensation value for the assigned units of channel 1 to 0.1 volts.

```
10  OUTPUT 707;":CHANNEL1:UNITS:OFFSET 0.1"
20  END
```

Query

```
:CHANnel<number>:UNITs:OFFSet?
```

This query returns the current units offset value for the specified channel.

Returned Format

```
[:CHANnel<number>:UNITs:OFFSet] <offset>
```

Example

The following example places the current unit offset value in the number variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":CHANNEL1:UNITS:OFFSET?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

14

Disk Commands

Disk Commands

The DISK subsystem commands perform the disk operations as defined under the disk menu. This allows storage and retrieval of waveforms, setups and pixel memory as well as formatting the disk.

The filenames used for files are compatible with DOS. They consist of up to 8 characters for the name with a 0 to 3 character extension separated by a '.' (dot). File names are all uppercase but can be entered in either upper or lower case and will be forced to upper case internally. Valid characters are [A-Z, 0-9, _ (the underscore character)].

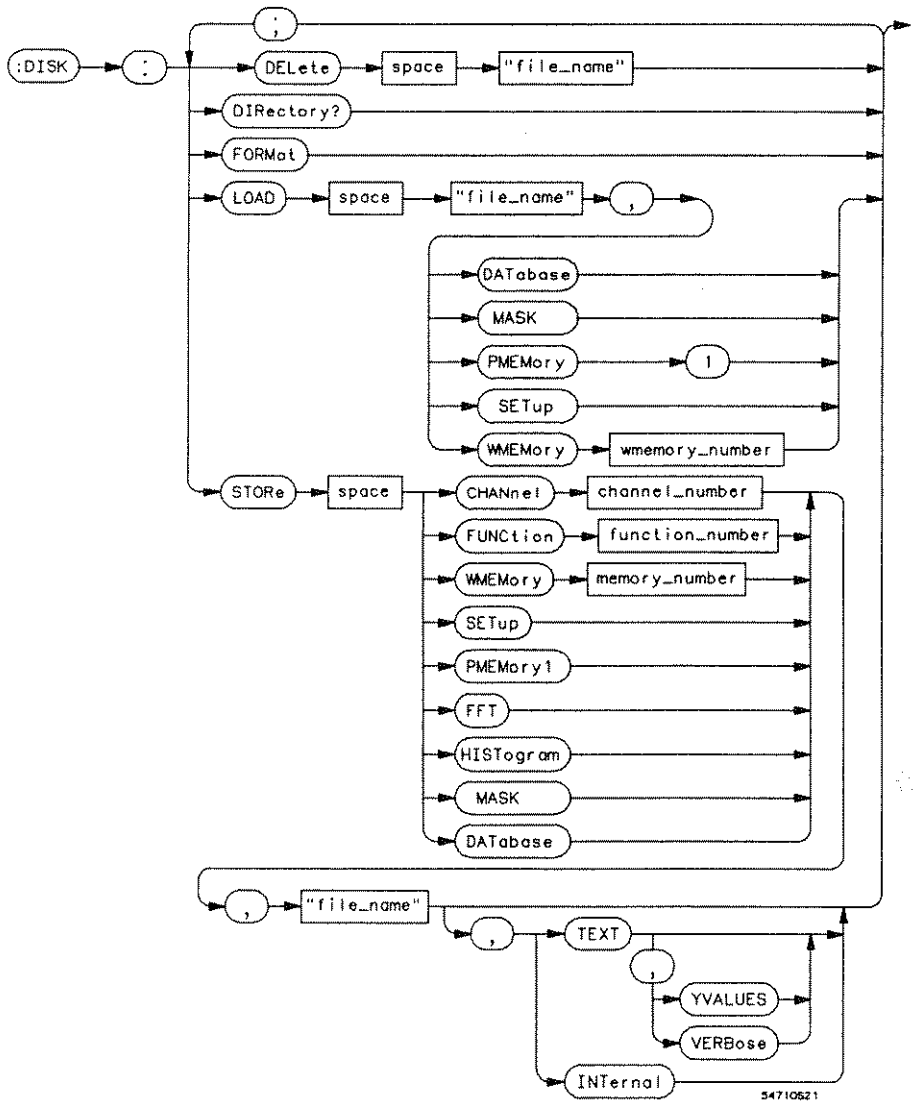
The filename must be enclosed in quotes.

The Disk subsystem contains the following commands and queries:

- DELete,
- DIRectory?,
- FORMat,
- LOAD, and
- STORe.

Figure 14-1 is the syntax diagram of the Disk Subsystem.

Figure 14-1



Disk Subsystem Syntax Diagram

DELete

Command

```
:DISK:DELete "<filename>"
```

The DISK:DELete command deletes a file from the disk. An error is displayed on the screen if the requested file does not exist.

`<filename>` A DOS compatible filename. Up to 8 characters with 3 character extension.

Example

```
10 OUTPUT 707;":DISK:DELETE ""FILE1.SET"""
20 END
```

DIRectory?

Query

```
:DISK:DIRectory?
```

This query returns the directory listing of the currently installed disk. Each entry is 63 bytes long including a carriage return and line feed.

Returned Format

```
[:DISK:DIRectory] <directory>
```

`<directory>` List containing <file1.ext> <type> <size> <date> <time> for each entry.

`<type>` One of {SETUP | WAVEFORM | WF_TEXT | PIXEL | TIFF | GIF | SUMMARY | PCX | PCL | EPSON | MASK | DATABASE | DOS | ASCII | IBPRG}

`<size>` Is an integer.

`<date>` In the format DD MMM YY

`<time>` In the format HH:MM

FORMat

Command

```
:DISK:FORMat
```

The DISK:FORMat command formats a disk in the drive. It is assumed that the disk that is to be formatted is in the drive when the command is issued.

Example

```
10 OUTPUT 707;":DISK:FORMAT"
20 END
```

LOAD

Command

```
:DISK:LOAD "<filename>"[,<destination>]
```

The DISK:LOAD command restores a setup, a waveform, a mask, a database, or a pixel memory from the disk. The type of file is determined by the filename suffix if one is present or by the destination field if one is not present. The database may only be loaded in internal format.

`<filename>` DOS compatible filename. Up to 8 characters with 0 to 3 character extension. Either .wav, .msk, .wdb, .txt, .pix, or .set may be used as a suffix after the filename. For a database, the file suffix must be .wdb. If no file suffix is specified, the default is .wav.

`<destination>` One of {WMEMoryN | SETup | PMEMory1 | DATAbase | MASK}

The maximum size of .txt waveforms is 128K points. Up to two waveform memories may be used when loading in a .txt file for records exceeding 64K points. Up to four waveform memories can be used if the file is in internal format.

Example

```
10 OUTPUT 707;":DISK:LOAD ""FILE1.WAV"",WMEM1"
20 END
```

STORe

Command

```
:DISK:STORe <source>, "<filename>"[,<format>]
```

The DISK:STORe command stores a setup, a waveform, a database, a mask,or pixel memory to the disk. The filename does not include a suffix. The suffix is supplied by the instrument depending on the source and file format specified. The database may only be saved in internal format.

`<source>` {CHANnel<number> | FUNCtionN<number> | WMEMory<number> | SETup | PMEMory1 | HISTogram | DATabase | FFT | MASK}

`<number>` For channels: 1, 2, 3, or 4, optionally followed by A or B.
For functions: 1 or 2.
For waveforms: 1, 2, 3, or 4.

`<filename>` Is an MS-DOS compatible name of the file up to 8 characters long

`<format>` One of {TEXT [,<YVALues> | <VERBose>] | INTernal}

> The format field is for waveforms, the default is INTernal. In TEXT mode, y values may be specified so that only the y values are stored. VERBose is the default in which y values and the waveform preamble is stored. Only waveforms of 128K or less may be written to disk in the TEXT formats. See the WAVeform Commands chapter for information on converting data to values.

Example

```
10 OUTPUT 707;":DISK:STORE SET,""FILE1.SET"""
20 END
```

15

Display Commands

Display Commands

The DISPLAY subsystem controls the display of data, markers, text, graticules, and the use of color.

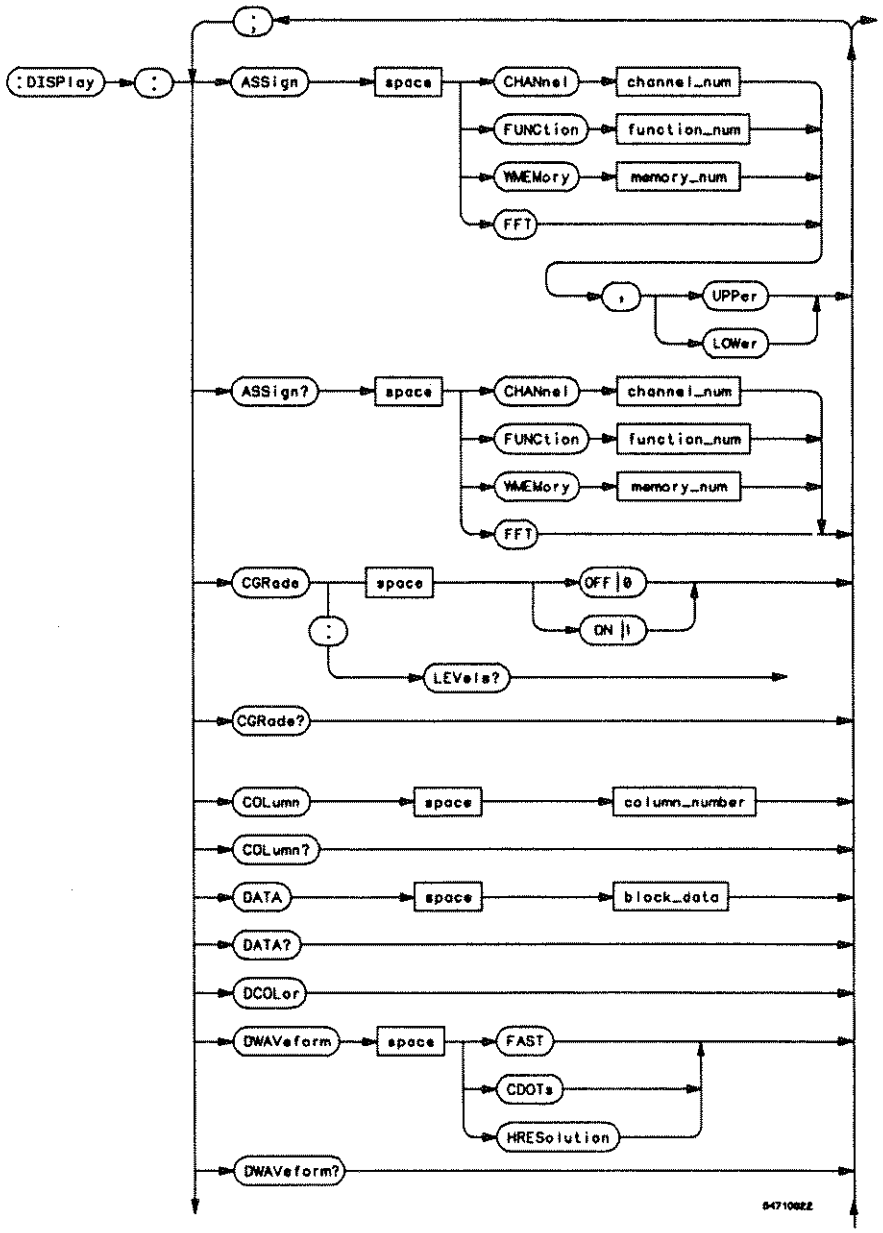
The display mode is selected by the ACQUIRE:TYPE command and the number of averages is selected by the ACQUIRE:COUNT command.

The Display subsystem contains the following commands:

- ASSign,
- CGRade,
- CGRade:LEVels,
- COLumn,
- DATA,
- DCOLor (Default COLor),
- DWAVeform (Draw WAVeform),
- FORMat,
- GRATicule,
- INVerse,
- LINE,
- MASK,
- PERSistence,
- ROW,
- SCOLor (Set COLor),
- SOURce,
- STRing, and
- TEXT.

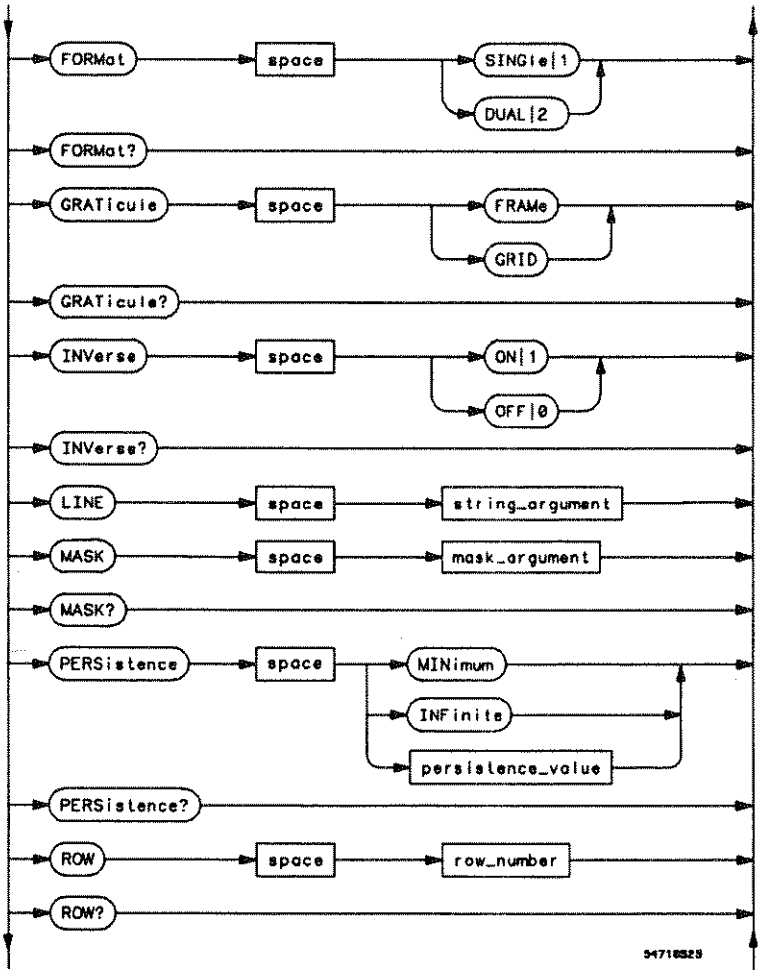
Figure 15-1 is the syntax diagram for the Display subsystem commands.

Figure 15-1



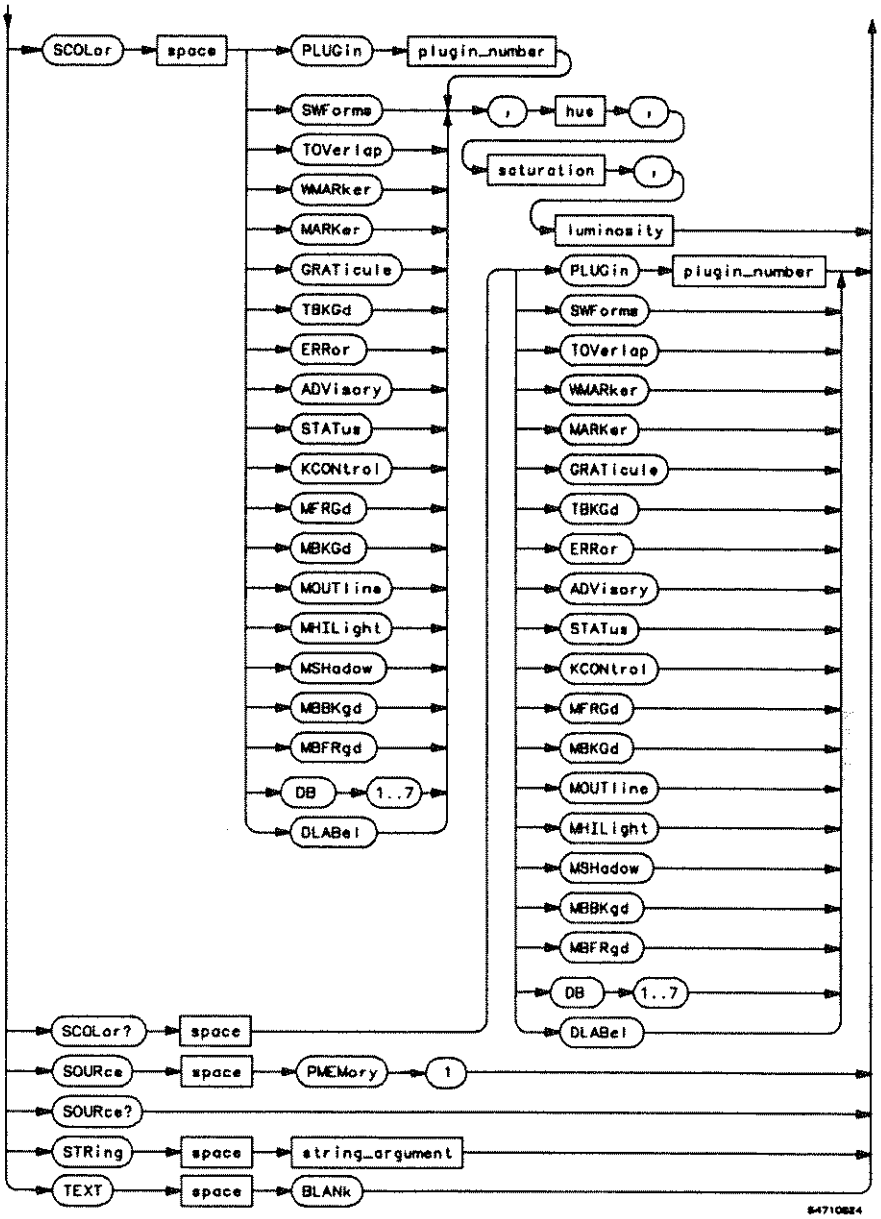
Display Subsystem Syntax Diagram

Figure 15-1 (continued)



Display Subsystem Syntax Diagram (continued)

Figure 15-1 (continued)



Display Subsystem Syntax Diagram (continued)

channel_num	an integer, 1 through 4.
function_num	an integer, 1 or 2.
memory_num	an integer, 1 or 2.
column_number	an integer, 0 through 81.
mask_argument	an integer, 0 to 255.
persistence_value	a real number, 0 to 40.
row_number	an integer, 0 to 27.
plugin_number	an integer, 1 through 4.
color_name	the name of a display color. See SCOLor for a list of color names.
string_argument	any series of ASCII characters enclosed in quotes.

Display Subsystem Commands Syntax Diagram (continued)

ASSign

Command

```
:DISPlay:ASSign {CHANnel<number> |
FUNCtion<number> | WMEMory<number> | FFT},{UPPer |
LOWer}
```

This command assigns the specified waveform, function, or FFT to the specified portion of the waveform area on the screen. This command has no effect when the graticule format is single.

`<number>` For channels: 1 through 4, optionally followed by an A or B.
For functions: 1 or 2.
For waveform memories: 1 to 4.

Example

The following example assigns channel 1 to the upper portion of the oscilloscope screen.

```
10  OUTPUT 707;":DISPLAY:ASSIGN CHANNEL1,UPPER"
20  END
```

Query

```
:DISPlay:ASSign? {CHANnel<number> |
FUNCtion<number> | WMEMory<number> |FFT}
```

The query returns the portion of the waveform area where the specified waveform is displayed or was last displayed if it is not currently on screen.

Returned Format

```
[:DISPlay:ASSign? {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | FFT} {UPPer | LOWer)<NL>
```

Example

The following example returns the portion of the waveform area where the channel 1 waveform resides to the string variable, Setting$, then it prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:ASSIGN? CHANNEL1"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

CGRade

Command

```
:DISPlay:CGRade {{ON | 1} | {OFF | 0}}
```

The DISPlay:CGRade command sets the color-graded display on or off. When in the color-graded display mode, all signals are mapped into a database and shown with different colors representing varying number of hits in a pixel. "Connected dots" and "high-resolution" display modes are disabled when the color-graded display is on.

The oscilloscope has three features that use a specific database. This database uses a different memory area than the waveform record for each channel. The three features that use the database are histograms, mask testing, and color-graded display. When any one of these three features is turned on, the oscilloscope starts building the database. The database is the size of the graticule area, which is 256 pixels high by 451 pixels wide. Behind each pixel is a 16-bit counter. Each counter is incremented each time a pixel is hit by data from a channel or function. The maximum count (saturation) for each counter is 63,488. You can check to see if any of the counters is close to saturation by using the DISPlay:CGRade ON command to enable the color-graded display feature. The color-graded display uses colors to represent the number of hits on various areas of the display.

The default color-grade state is off.

Example

The following example sets the color-graded display on.

```
10  OUTPUT 707;":DISPLAY:CGRADE ON"
20  END
```

Query

```
:DISPlay:CGRade?
```

The DISPlay:CGRade query returns the current color-grade state.

Returned Format

```
[:DISPlay:CGRade] {ON | OFF} <NL>
```

Example

The following example returns the current color grade state.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:CGRADE?"
30  ENTER 707;Cgrade$
40  PRINT Cgrade$
50  END
```

CGRade:LEVels?

Query

```
:DISPlay:CGRade:LEVels?
```

The DISPlay:CGRade:LEVels query returns the range of hits represented by each color. Fourteen values are returned, representing the minimum and maximum count for each of seven colors. The values are returned in the following order:

Greatest intensity color minimum,
greatest intensity color maximum,
next greatest intensity color minimum,
next greatest intensity color maximum,
...
least intensity color minimum, and
least intensity color maximum.

Returned Format

```
[:DISPlay:CGRade:LEVels] <color format> <NL>
```

<color format> <intensity color min / max> is an integer value from 0 to 65535

Example

The following example gets the range of hits represented by each color and prints it on the controller screen:

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:CGRADE:LEVELS?"
30  ENTER 707;Cgrade$
40  PRINT Cgrade$
50  END
```

COLumn

Command

```
:DISPlay:COLumn <value>
```

This command specifies the starting column for subsequent DISPlay:STRing and DISPlay:LINE commands.

`<value>` An integer, 0 to 81, representing the starting column for subsequent STRING and LINE commands. The entire viewing area of the screen is divided into 28 rows by 82 columns, so your text can be written anywhere on the screen.

Example

The following example sets the starting column for subsequent DISPlay:STRing and DISPlay:LINE commands to column 10.

```
10  OUTPUT 707;":DISPLAY:COLUMN 10"
20  END
```

Query

```
:DISPlay:COLumn?
```

The DISPlay:COLumn query returns the column where the next DISPlay:LINE or DISPlay:STRing starts.

Returned Format

```
[:DISPlay:COLumn] <value><NL>
```

Example

The following example returns the current column setting to the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:COLUMN?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

DATA

Command

```
:DISPlay:DATA <binary_block_data>
```

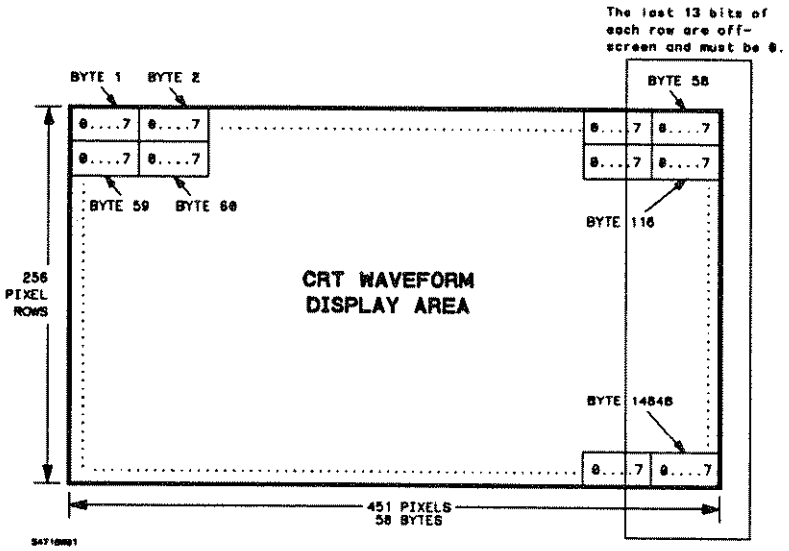
This command writes waveform data to the pixel memory of the instrument (PMEMory1). The DATA command is followed by a block of binary data that is transferred from the controller to the pixel memory in the instrument.

<Binary_block_data> Data in the IEEE 488.2 definite block format.

Pixel Format

The data that is output by this command is the displayed data from the waveform display area (inside the graticule) of the oscilloscope CRT. When using the DISPLAY:DATA command, the information is output as bytes of pixel data. The waveform display area is 451 pixels horizontally by 256 pixels vertically. Each pixel row is divided into 58 bytes of eight bits each as in figure 15-2.

Figure 15-2



CRT Pixel Data

Data Output Format

The pixel plane is 14,848 bytes representing the waveform display portion of the display. The waveform display portion is from and including the top graticule line, to and including the bottom graticule line, as well as all information inside the graticule. The first 58 bytes of data that are output correspond to the top row of the waveform display. The next 58 bytes are the second row of the waveform display, the third 58 bytes correspond to the third row, and so on. The data is output in rows until the 256th pixel row of data is sent. The 256th row corresponded to the bottom graticule line of the waveform display area.

Each of the pixel rows contain 451 pixels that are sent in 8-bit bytes, and 256 rows of pixel data are sent. Each row is sent as 58 bytes. There are 13 unused bits in each row, because 58 bytes times 8 bits equals 464 bits per row. 256 rows of data are sent, so 58 bytes times 256 rows equals 14,848 bytes sent for each screen of data.

The 13 unused bits in each row are off-screen, and must be set to zero when sending data to display.

Query

```
:DISPlay:DATA?
```

Returned Format

```
[:DISPlay:DATA] <binary_block_data><NL>
```

The query returns waveform data from the pixel memory of the instrument. The query causes the instrument to output pixel data from the pixel memory. The query also follows the IEEE 488.2 definite block format.

This oscilloscope has one pixel memory.

DCOLor (Default COLor)

Command

```
:DISPlay:DCOLor
```

This command (Default COLor) returns the screen colors to the predefined colormap.

Example

This example sends the DCOLor command.

```
10  OUTPUT 707;":DISPLAY:DCOLOR"
20  END
```

DWAVeform (Draw WAVeform)

Command

```
:DISPlay:DWAVeform {FAST | CDOTs | HRESolution}
```

This command (Draw WAVeform) sets the waveform draw mode to FASt, Connected DOTs, or High RESolution. This command has no effect when the color-graded display is on.

Example

This example sets the waveform draw mode to high resolution.

```
10  OUTPUT 707;":DISPLAY:DWAVEFORM HRESOLUTION"
20  END
```

Query

```
:DISPlay:DWAVeform?
```

The query returns the current waveform draw mode.

Example

The following example places the current setting for the waveform draw mode in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                    !Dimension variable
20  OUTPUT 707;":DISPLAY:DWAVEFORM?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FORMat

Command

```
:DISPlay:FORMat {{SINGle | 1} | {DUAL | 2}}
```

The DISPlay:FORMat command sets the number of graphs. Sending 1 or SINGle sets number of graphs to 1. Sending 2 or DUAL sets the number of graphs to 2. Each graph defines a separate graticule area within the waveform display area.

Eight divisions are used for the full scale range in both format modes. Waveforms can be assigned to each area of the screen with the DISPLAY:ASSIGN command.

Example

The following example sets the number of graphs to 2.

```
10  OUTPUT 707;":DISPLAY:FORMAT DUAL"
20  END
```

Query

```
:DISPlay:FORMat?
```

The DISPlay:FORMat query returns the current number of display areas on the screen.

Returned Format

```
[:DISPlay:FORMat] {1 | 2}<NL>
```

Example

The following example places the current setting for the display format in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:FORMAT?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

GRATicule

Command

```
:DISPlay:GRATicule {GRID | FRAMe}
```

This command selects the type of graticule that is displayed.

Example

The following example sets up the background of the oscilloscope's display with a frame separated into major and minor divisions.

```
10  OUTPUT 707;":DISPLAY:GRATICULE FRAME"
20  END
```

Query

```
:DISPlay:GRATicule?
```

The DISPlay:GRATicule query returns the type of graticule currently displayed.

Returned Format

```
[:DISPlay:GRATicule] {GRID | FRAMe}<NL>
```

Example

The following example places the current setting of the display graticule in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:GRATICULE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

INVerse

Command

```
:DISPlay:INVerse {{ON | 1} | {OFF | 0}}
```

This command determines whether or not text sent with the DISPlay:LINE or DISPlay:STRing command will be written with the INVERSE attribute. If the inverse attribute is on, the text is written in inverse video.

Example

The following example turns the inverse attribute on for future DISPlay:LINE and DISPlay:STRing commands.

```
10  OUTPUT 707;":DISPLAY:INVERSE ON"
20  END
```

Query

```
:DISPlay:INVerse?
```

The DISPlay:INVerse query returns the current state of the DISPlay:INVerse command.

Returned Format

```
[:DISPlay:INVerse] {1 | 0}<NL>
```

Example

The following example places the current setting of the inverse attribute in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:INVERSE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

LINE

Command

```
:DISPlay:LINE "<string>"
```

This command writes a quoted string parameter to the screen, starting at the location specified by the DISPlay:ROW and DISPlay:COLumn commands.

`<string>` Any series of ASCII characters enclosed in quotes.

Example

The following example writes the message "Test 1" to the screen, starting at the current row and column location.

```
10  OUTPUT 707;":DISPLAY:LINE ""TEST 1"""
20  END
```

Text may be written up to column 81. If the characters in the string parameter do not fill the line, the rest of the line is blanked. If the string is longer than the space available on the current line, the excess characters are discarded. In any case, the ROW is incremented and the COLumn remains the same. The next DISPlay:LINE command will write on the next line of the display. After writing line 27, the last line in the display area, the ROW is reset to 0.

MASK

Command

```
:DISPlay:MASK <mask_value>
```

The DISPlay:MASK command inhibits the instrument from writing to selected areas of the screen. The mask parameter is an 8-bit integer in which each bit controls writing to an area of the screen. A 0 inhibits writing to the area represented by the bit, a 1 enables writing to that area.

<mask_value> An integer, 0 to 255, representing the areas of the screen that will be masked off to inhibit the instrument from writing to them (see table 14-1).

Example

The following example inhibits the menu area (bit 6) from being written to by the instrument.

```
10  OUTPUT 707;":DISPLAY:MASK 191"
20  END
```

Query

```
:DISPlay:MASK?
```

The DISPlay:MASK query returns the current value of the mask.

Returned Format

```
[:DISPlay:MASK] <mask_value>
```

<mask_value> An integer, 0 to 255, representing the areas of the screen that are masked off to inhibit the instrument from writing to them (see table 15-1).

Example

The following example returns the current value of the mask to the string variable, Value$, then prints the contents of the variable to the controller's screen.

```
10  DIM Value$[50]                    !Dimension variable
20  OUTPUT 707;":DISPLAY:MASK?"
30  ENTER 707;Value$
40  PRINT Value$
50  END
```

Text sent with the DISPlay:LINE and DISPlay:STRing commands, or with the SYSTem:DSP command is not affected by this command.

The purpose of the command is to allow HP-IB text to be written anywhere on screen and to prevent the instrument from overwriting the text through its normal operation.

When an area that has been masked off is unmasked, the instrument immediately redraws that area with the information that would normally appear in that area. If any area of the screen is masked off, then full screen text windows, such as Help, System Configuration, and Calibration status will not be displayed when they are selected (such as with the MENU:HELP command).

The DISPlay:MASK parameters are not reset with a *RST common command. The mask value is reset to 255 whenever a remote-to-local transition occurs.

Table 15-1 **Display Byte Mask**

Bit	Weight	Screen Area Affected
7	128	Unused
6	64	Menu Area (right side)
5	32	Time Base Scale Area (graticule frame)
4	16	Measurement Area (bottom five lines)
3	8	Graticule Area
2	4	Unused
1	2	Status Lines (second row)
0	1	Advisory Area (message location, top row)

PERSistence

Command

```
:DISPlay:PERSistence {MINimum | INFinite | <value>}
```

This command sets the display persistence. It works in both real and equivalent time. The parameter for this command can be either MINIMUM (zero persistence), INFINITE, or a real number from 0 to 40.0, representing the persistence in seconds.

`<value>` A real number, 0 to 40.0, representing the persistence in seconds.

Example

The following example sets the persistence to infinite.

```
10  OUTPUT 707;":DISPLAY:PERSISTENCE INFINITE"
20  END
```

Query

```
:DISPlay:PERSistence?
```

The :DISPlay:PERSistence query returns the current persistence value.

Returned Format

```
[:DISPlay:PERSistence] {MINimum | INFinite | <value>}<NL>
```

Example

The following example places the current persistence setting in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:PERSISTENCE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

ROW

Command

```
:DISPlay:ROW <row_number>
```

This command specifies the starting row on the screen for subsequent DISPlay:STRing and DISPlay:LINE commands. The row number remains constant until another DISPlay:ROW command is received, or the row is incremented by the DISPlay:LINE command.

`<row_number>` An integer, 0 to 27, representing the starting row for subsequent DISPlay:STRing and DISPlay:LINE commands. The entire screen viewing area is divided into 28 rows by 82 columns, so your text can be written anywhere on the screen.

Example The following example sets the starting row for subsequent DISPlay:STRing and DISPlay:LINE commands to 10.

```
10  OUTPUT 707;":DISPLAY:ROW 10"
20  END
```

Query

```
:DISPlay:ROW?
```

The DISPlay:ROW query returns the current value of the row.

Returned Format

```
[:DISPlay:ROW] <row_number><NL>
```

Example The following example places the current value for row in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":DISPLAY:ROW?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

SCOLor

Command

```
:DISPlay:SCOLor <color_name>, <hue>, <saturation>,
<luminosity>
```

<color_name>

```
{PLUGin1 | PLUGin2 | PLUGin3 | PLUGin4 | SWForms |
TOVerlap | WMARker | MARKer | GRATicule | TBKGd |
ERRor | ADVisory | STATus | KCONtrol | MFRGd |
MBKGd | MOUTline | MHILight | MSHadow | MBBKgd |
MBFRgd | DLABel | DB1..DB7}
```

The DISPlay:SCOLor command changes the colors used on the display of the oscilloscope. The :DISPlay:DCOLor command restores the colors to their factory default settings.

The database colors, DB1..DB7, can be changed only when the color-graded display is on. See the DISPlay:CGRade command.

The color names are defined as follows:

Table 15-2 Color Names

Color Name	Definition
PLUGin1	Used for waveforms acquired from slot 1 and for the function 1 waveform.
PLUGin2	Used for waveforms acquired from slot 2 and for the function 2 waveform.
PLUGin3	Used for waveforms acquired from slot 3 and for the FFT waveform.
PLUGin4	Used for waveforms acquired from slot 4.
SWForms	Saved waveforms. Used for waveform and pixel memories.
TOVerlap	The trace overlap color. Shown where waveforms of different colors overlap.
WMARker	Timebase window marker. Also used for waveform masks and waveform mask coordinate system markers.
MARKer	General purpose markers (waveform markers, trigger level, histograms, and so on).
GRATicule	Waveform area graticule.
TBKGd	Trace background color. The background color in the graticule area.

Color Name	Definition
ERRor	Error messages. Typically used for messages displayed above the graticule area.
ADVisory	Advisory messages. Used for text displayed above the graticule area.
STATus	Status messages displayed above the graticule.
KCONtrol	Knob control indicator color. Used to indicate control to which knob is slaved.
MFRGd	Menu foreground.
MBKGd	Menu background and graticule border.
MOUTline	Menu outline and screen background outside the graticule area.
MHILight	Menu highlight color and edges of graticule border.
MSHadow	Menu shadow color. Used to give menu graphics a 3-dimensional look.
MBBKgd	Memory bar background color.
MBFRgd	Memory bar foreground color. The color of the memory bar above the graticule.
DLABel	User-defined display labels
DB1..DB7	The seven ranges of pixel counts for the color-graded display.

`<hue>` An integer, 0 to 100, that determines the graduation of color. As the hue number increases, the selected color cycles through the color spectrum. There is no difference between hue 0 and hue 100.

`<saturation>` An integer, 0 to 100, that selects the percentage of the pure color that gets mixed with white. Zero is white and 100 is the maximum saturation of the selected color.

`<luminosity>` An integer, 0 to 100, that determines the brightness of the selected hue. Zero is black and 100 is the maximum brightness.

Example

The following example sets the hue to 50, the saturation to 70, and the luminosity to 90 for the markers..

```
10  OUTPUT 707;":DISPLAY:SCOLOR MARKER,50,70,90"
20  END
```

Query

```
:DISPlay:SCOLor? <color_name>
```

The DISPlay:SCOLor query returns the hue, saturation, and luminosity for the specified color.

Returned Format

```
[:DISPlay:SCOLor] <color_name>, <hue>, <saturation>,
<luminosity><NL>
```

Example

The following example places the current settings for the graticule color in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":DISPLAY:SCOLOR? GRATICULE"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

SOURce

Command

```
:DISPlay:SOURce PMEMory1
```

This command specifies the destination or source for the DISPlay:DATA command and query. Pixel memory 1 (PMEMORY1) is the only source available for this command.

Example

The following example selects pixel memory 1 as the display source.

```
10  OUTPUT 707;":DISPLAY:SOURCE PMEMORY1"
20  END
```

Query

```
:DISPlay:SOURce?
```

The query always returns PMEMory1.

Returned Format

```
[:DISPlay:SOURce] PMEMory1<NL>
```

Example

The following example places the current selection for the display source in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":DISPLAY:SOURCE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

STRing

Command

```
:DISPlay:STRing <string>
```

The DISPlay:STRing command writes text to the screen of the oscilloscope. The text is written starting at the current row and column settings. If the column limit is reached (81), the excess text is discarded. The DISPlay:STRing command does not increment the row value; however, the DISPlay:LINE command does.

`<string>` Any series of ASCII characters enclosed in quotes.

Example

The following example writes the message "Example 1" to the oscilloscope's display starting at the current row and column settings.

```
10  OUTPUT 707;":DISPLAY:STRING ""Example 1"""
20  END
```

TEXT

Command

```
:DISPlay:TEXT BLANk
```

The DISPlay:TEXT command blanks the user text area of the screen. The user text area includes rows 0 through 27, columns 0 through 81.

Example

The following example blanks the user text area of the oscilloscope's screen.

```
10  OUTPUT 707;":DISPLAY:TEXT BLANK"
20  END
```

16

Function Commands

Function Commands

The FUNCTION subsystem defines two functions, function 1 and function 2. The operands of these two functions can be any installed channels in the oscilloscope, waveform memories 1 through 4, functions 1 or 2, or a constant.

The Function subsystem contains the following commands and queries:

- ADD,
- DIFFerentiate,
- DISPlay,
- DIVide,
- FFT:FREQuency,
- FFT:MAGNify,
- FFT:MSPan,
- FFT:RESolution,
- FFT:SPAN,
- FFT:WINDow,
- FFTMagnitude,
- HORizontal,
- HORizontal:POSition,
- HORizontal:RANGe,
- INTegrate,
- INVert,
- MAGNify,
- MAXimum,
- MINimum,
- MULTiply,
- OFFSet,
- ONLY,
- RANGe,
- SUBTract,
- VERSUS, and
- VERTical.

The vertical scaling and offset functions can be controlled remotely using the RANGE and OFFSET commands in this subsystem. The horizontal scaling and delay values of the functions can be obtained using the XRANGE and XPOSITION queries in this subsystem.

If a channel is not on and is specified as an operand, then that channel is acquired.

If the operand waveforms have different record lengths, then the function takes the shorter of the two.

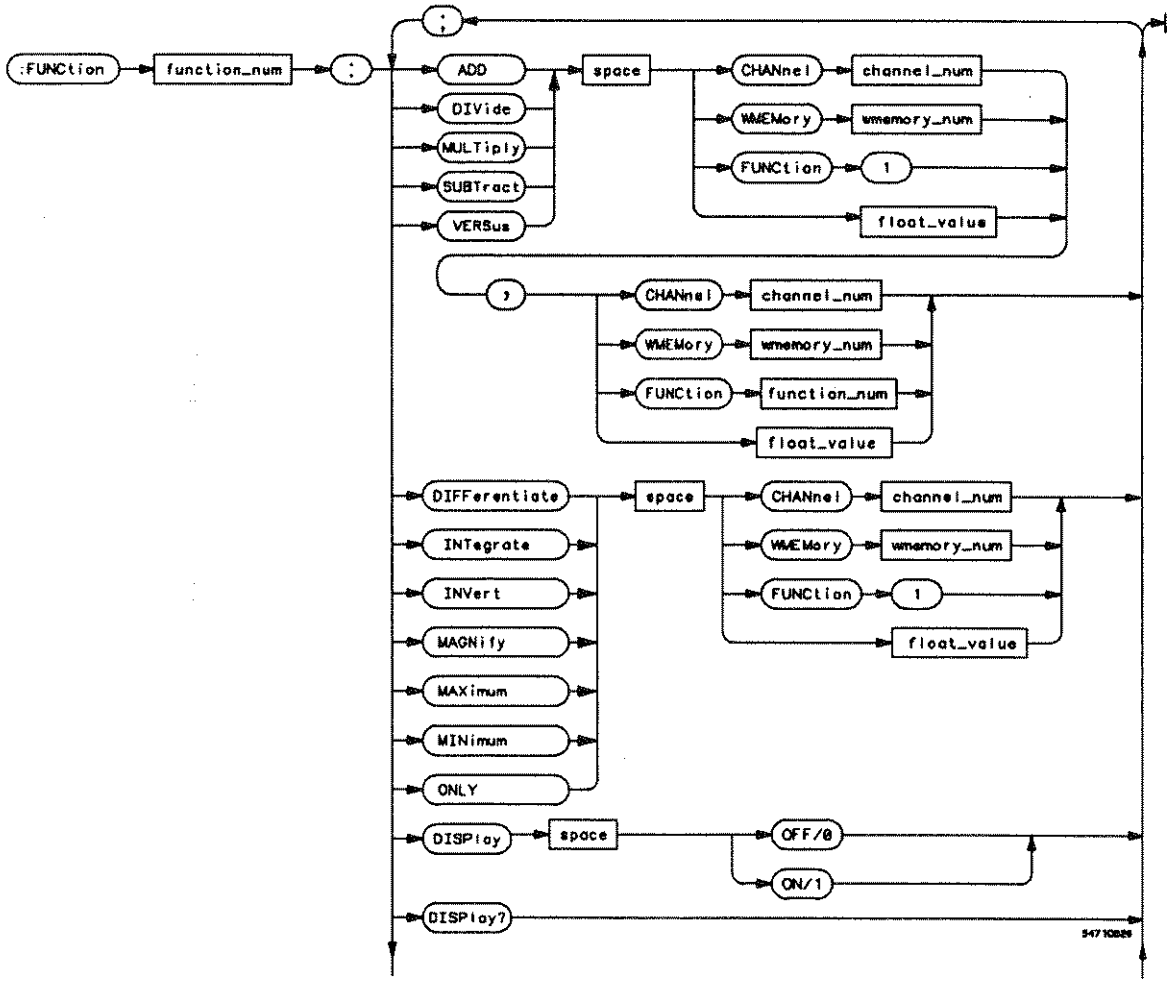
If the two operands have the same time scales, then the resulting function has the same time scale. If the operands have different time scales, then the resulting function has no valid time scale, since operations are done based on waveform record position and the time relationship of the data records is not considered. When the time scale is not valid, delta time pulse parameter measurements have no meaning and the unknown result indicator is displayed on screen.

Constant operands take on the same time scale as the associated waveform operand.

The maximum length of a waveform record is determined by the combination of plug-in and oscilloscope frame. See the ACQuire:POINts command. However, all functions are limited to the first 32K points of the waveform record that are on screen. This will not necessarily be the first 32K points of the full record unless all of those points are on screen.

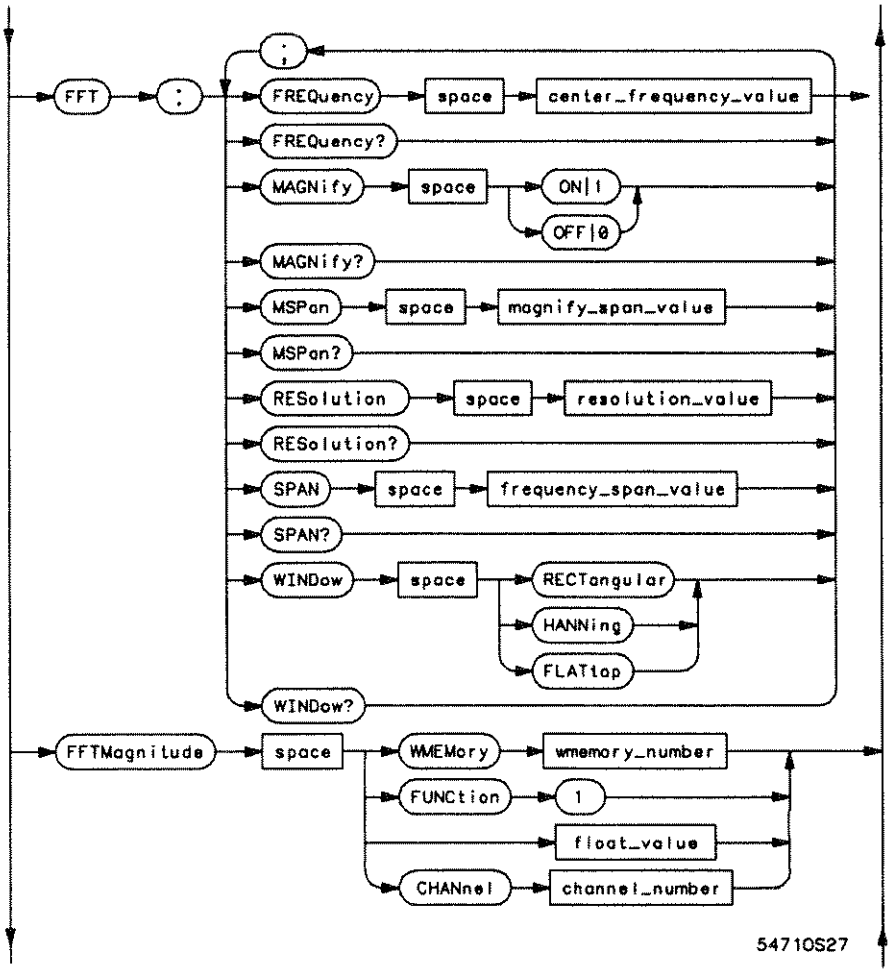
Figure 16-1 is the syntax diagram for the Function subsystem commands.

Figure 16-1



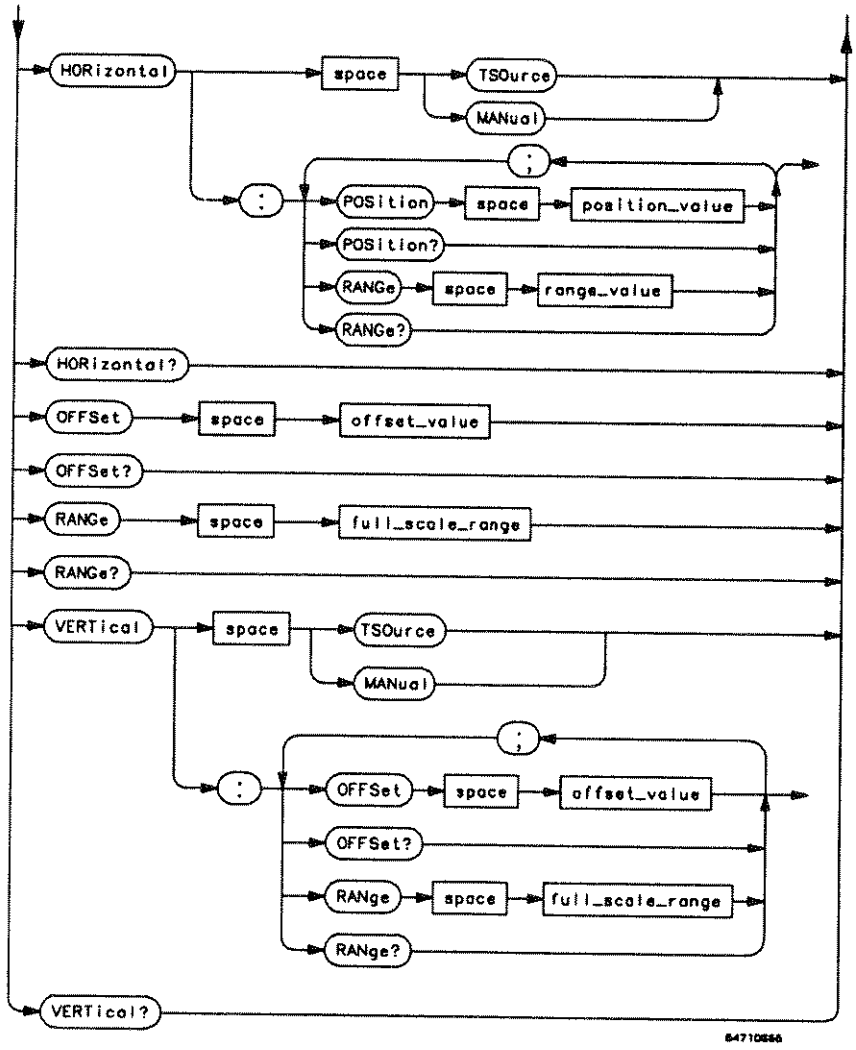
Function Subsystem Syntax Diagram

Figure 16-1 (continued)



Function Subsystem Syntax Diagram (continued)

Figure 16-1 (continued)



Function Subsystem Syntax Diagram (continued)

position_value	time of horizontal position
full_scale_range	time or voltage range
offset_value	vertical offset in the currently selected Y-axis units (normally volts
channel_num	1, 2, 3, or 4
wmemory_num	1, 2, 3, or 4
function_num	1 or 2
center_frequency_value	Frequency in Hertz.
magnify_span_value	Frequency in Hertz.
resolution_value	Frequency in Hertz.
frequency_span_value	Frequency in Hertz.
float_value	a real number

Function Subsystem Syntax Diagram (continued)

ADD

Command

```
:FUNCtion<number>:ADD <operand>,<operand>
```

The FUNCtion<number>:ADD command defines a function that takes the algebraic sum of two defined operands.

<number> An integer, 1 or 2, representing the selected function.

<operand> {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

> Function 2 may use Function 1 as an operand, but Function 1 may not use Function 2 as an operand.

Example

The following example sets up function 1 to add channel 1 to channel 2.

```
10  OUTPUT 707;":FUNCTION1:ADD CHANNEL1,CHANNEL2"
20  END
```

DIFFerentiate

Command

```
:FUNCtion<number>:DIFF <operand>
```

The FUNCtion<number>:DIFFerentiate command defines a function that computes the discrete derivative of the defined operand's waveform.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to take the discrete derivative of the signal on channel 2.

```
10  OUTPUT 707;":FUNCTION2:DIFFERENTIATE CHANNEL2"
20  END
```

DISPlay

Command

```
:FUNCtion<number>:DISPlay {{ON | 1} | {OFF | 0}}
```

The FUNCtion<number>:DISPlay command turns the selected function on and off.

`<number>` An integer, 1 or 2, representing the selected function.

Example The following example turns function 1 on.

```
10  OUTPUT 707;":FUNCTION1:DISPLAY ON"
20  END
```

Query

```
:FUNCtion<number>:DISPlay?
```

The FUNCtion<number>:DISPlay query returns the state of the specified function.

Returned Format

```
[:FUNCtion<number>:DISPlay] {1 | 0}<NL>
```

Example The following example places the current state of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":FUNCTION1:DISPLAY?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

DIVide

Command

```
:FUNCtion<number>:DIVide <operand>,<operand>
```

The FUNCtion<number>:DIVide command defines a function that divides the first operand by the second operand.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to divide the signal on channel 1 by the signal in waveform memory 4.

```
10  OUTPUT 707;":FUNCTION2:DIVIDE CHANNEL1,WMEMORY4"
20  END
```

FFT:FREQuency

Command

```
:FUNCtion<number>:FFT:FREQuency
<center_frequency_value>
```

This command sets the center frequency for the FFT when magnify is on. Magnify is turned on and off with the :FUNCtion<number>:FFT:MAGNify command.

`<number>` Function number 1 or 2.

`<center_frequency_value>` Value in Hertz.

Query

```
:FUNCtion<number>:FFT:FREQuency?
```

The query returns the center frequency value when magnify is on.

Returned Format

```
[FUNCtion<number>:FFT:FREQuency] <center_frequency_value><NL>
```

FFT:MAGNify

Command

```
:FUNCtion<number>:FFT:MAGNify{{ON | 1} | {OFF | 0}}
```

This command enables the magnification mode.

`<number>` Function number 1 or 2.

Query

```
:FUNCtion<number>:FFT:MAGNify?
```

The query returns the current state of the magnification mode.

Returned Format

```
[FUNCtion<number>:FFT:MAGNify] {1 | 0}<NL>
```

FFT:MSPan

Command

```
:FUNCtion<number>:FFT:MSPan
<magnify_span_value><NL>
```

`<number>` Function number 1 or 2.

`<magnify_span_value>` Span frequency in Hertz.

This command sets the span when magnify is on. Magnify is turned on and off with the :FUNCtion:FFT:MAGNify command.

Query

```
:FUNCtion<number>:FFT:MSPan
```

The query returns the current span value when magnify is on.

Returned Format

```
[:FUNCtion:FFT:MSPan] <magnify_span_value><NL>
```

FFT:RESolution

Command

```
:FUNCtion<number>:FFT:RESolution <resolution_value>
```

`<number>` Function number 1 or 2.

`<resolution_value>` Resolution value in Hertz.

This command sets the resolution (number of points) for the FFT function.

Query

```
:FUNCtion<number>:FFT:RESolution?
```

The query returns the current resolution value of the FFT function.

Returned Format

```
[FUNCtion:FFT:RESolution] <resolution_value><NL>
```

FFT:SPAN

Command `:FUNCtion<number>:FFT:SPAN <frequency_span_value>`

`<number>` Function number 1 or 2.

`<frequency_span_value>` Span value in Hertz.

This command sets the frequency span value for the FFT function in the unmagnified mode.

Query :FUNCtion:<number>FFT:SPAN?

The query returns the current frequency span value of the FFT function in the unmagnified mode.

Returned Format `[:FUNCtion:FFT:SPAN] <frequency_span_value><NL>`

FFT:WINDow

Command `:FUNCtion<number>:FFT:WINDow {RECTangular | HANNing | FLATtop}`

This command sets the window type for the FFT function.

The FFT function assumes that the time record repeats. Unless there is an integral number of cycles of the sampled waveform in the record, a discontinuity is created at the beginning of the record. This introduces additional frequency components into the spectrum about the actual peaks. This is referred to as spectral leakage. In order to minimize spectral leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs.

Each window is useful for certain classes of input signals.

The RECTangular window is essentially no window, all points are multiplied by 1. The RECTangular window is useful for transient signals and signals where there are an integral number of cycles in the time record. The HANNing window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making

frequency measurements. The FLATtop window is best for making accurate amplitude measurements of frequency peaks.

`<number>` Function number 1 or 2.

Example

```
10 OUTPUT 707;":FUNCtion<number>:FFT:WINDow RECTangular
20 END
```

Query

`:FUNCtion<number>:FFT:WINDow?`

The query returns the current selected window for the FFT function.

Returned Format

```
[:FUNCtion<number>:FFT:WINDow] {RECTangular | HANNing |
FLATtop}<NL>
```

Example

```
10 DIM WND$[50]
20 OUTPUT 707;":FUNCTION1:FFT:WINDOW?
30 ENTER 707;WND$
40 PRINT WND$
50 END
```

FFTMagnitude

Command

`:FUNCtion<number>:FFTMagnitude <operand>`

The FUNCtion<number>:FFTMagnitude command computes the fast Fourier transform of the specified channel or memory. The FFT takes the digitized time record of the specified channel or memory and transforms it to the frequency domain.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 1 to compute the FFT of waveform memory 3.

```
10 OUTPUT 707:":FUNCTION1:FFTMAGNITUDE WMEMORY3"
20 END
```

HORizontal

Command

```
:FUNCtion<number>:HORizontal {TSOurce | MANual}
```

The FUNCtion<number>:HORizontal command sets the horizontal tracking to either Track SOurce or MANual.

The HORizontal command also includes a subsystem consisting of the following commands and queries:

- POSition, and
- RANGe.

`<number>` An integer, 1 or 2, representing the selected function.

Query

```
:FUNCtion<number>:HORizontal?
```

The query returns the current horizontal scaling mode of the specified function.

Returned Format

```
[:FUNCtion<number>:HORizontal] {TSOurce | MANual}<NL>
```

Example

The following example places the current state of horizontal tracking of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":FUNCTION1:HORIZONTAL?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

HORizontal:POSition

Command

```
:FUNCtion<number>:HORizontal:POSition
<position_value>
```

The FUNCtion<number>HORizontal:POSition command sets the time value at center screen for the selected function.

`<number>` Function number 1 or 2.

`<position_value>` Position value in time.

Query

```
:FUNCtion<number>:HORizontal:POSition?
```

The query returns the current time value at center screen of the selected function.

Returned Format

```
[:FUNCtion<number>:HORizontal:POSition] <position><NL>
```

Example

The following example places the current horizontal position setting for function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"         !Response headers off
20  OUTPUT 707;":FUNCTION2:HORIZONTAL:POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

HORizontal:RANGe

Command

```
:FUNCtion<number>:HORizontal:RANGe <range_value>
```

The FUNCtion<number>HORizontal:RANGe command sets the current time range for the specified function.

`<number>` Function number 1 or 2.

`<range_value>` Width of screen in current x-axis units (usually seconds).

Query

```
:FUNCtion<number>:HORizontal:RANGe?
```

The query returns the current time range setting of the specified function.

Returned Format

```
[:FUNCtion<number>:HORizontal:RANGe] <range><NL>
```

Example

The following example places the current horizontal range setting of function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"        !Response headers off
20  OUTPUT 707;":FUNCTION2:HORIZONTAL:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

INTegrate

Command

```
:FUNCtion<number>:INTegrate <operand>
```

The FUNCtion<number>:INTegrate command defines a function that computes the integral of the defined operand's waveform.

<number> An integer, 1 or 2, representing the selected function.

<operand> {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 1 to compute the integral of waveform memory 3.

```
10  OUTPUT 707;":FUNCTION1:INTEGRATE WMEMORY3"
20  END
```

INVert

Command

```
:FUNCtion<number>:INVert <operand>
```

The FUNCtion<number>:INVert command defines a function that inverts the defined operand's waveform by multiplying by -1.

<number> An integer, 1 or 2, representing the selected function.

<operand> {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to invert the signal on channel 1.

```
10  OUTPUT 707;":FUNCTION2:INVERT CHANNEL1"
20  END
```

MAGNify

Command

```
:FUNCtion<number>:MAGNify <operand>
```

The FUNCtion<number>:MAGNify command defines a function that is a copy of the operand. The magnify function is a software magnify. No hardware settings are altered as a result of using this function. It is useful for scaling channels, another function, and memories with the RANGE and OFFSET commands in this subsystem.

Magnify performs the same operation as the "ONLY" operator and is the preferred operator of the two. "ONLY" is included in this instrument for compatibility with previous instruments.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example creates a function (function 1) that is a magnify copy of channel 1.

```
10  OUTPUT 707;":FUNCTION1:MAGNIFY CHANNEL1"
20  END
```

MAXimum

Command

```
:FUNCtion<number>:MAXimum <operand>
```

The FUNCtion<number>:MAXimum command defines a function that computes the maximum value of the operand waveform in each time bucket.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to compute the maximum of each time bucket for channel 2A.

```
10  OUTPUT 707;":FUNCTION2:MAXIMUM CHANNEL2A"
20  END
```

MINimum

Command

```
:FUNCtion<number>:MINimum <operand>
```

The FUNCtion<number>:MINimum command defines a function that computes the minimum of each time bucket for the defined operand's waveform.

<number> An integer, 1 or 2, representing the selected function.

<operand> {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example sets up function 2 to compute the minimum of each time bucket for channel 4B.

```
10  OUTPUT 707;":FUNCTION2:MINIMUM CHANNEL4B"
20  END
```

MULTiply

Commad

`:FUNCtion<number>:MULTiply <operand>,<operand>`

The FUNCtion<number>:MULTiply command defines a function that algebraically multiplies the first operand by the second operand.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example defines a function that multiplies channel 1 by waveform memory 1.

```
10  OUTPUT 707;":FUNCTION1:MULTIPLY CHANNEL1,WMEMORY1"
20  END
```

OFFSet

Command

```
:FUNCtion<number>:OFFSet <offset_value>
```

The FUNCtion<number>:OFFSet command sets the voltage represented at the center of the screen for the selected function. The offset is limited to being within the vertical range that can be represented by the function data.

`<number>` An integer, 1 or 2, representing the selected function.

`<offset_value>` The offset value is limited to being within the vertical range that can be represented by the function data.

Example

The following example sets the offset voltage for function 1 to 2 mV.

```
10  OUTPUT 707;":FUNCTION1:OFFSET 2E-3"
20  END
```

Query

```
:FUNCtion<number>:OFFSet?
```

The FUNCtion<number>:OFFSet query returns the current offset value for the selected function.

Returned Format

```
[:FUNCtion<number>:OFFSet] <offset_value><NL>
```

Example

The following example places the current setting for offset on function 2 in the numeric variable, Value, then prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:OFFSET?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

> When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

ONLY

Command

```
:FUNCtion<number>:ONLY <operand>
```

The FUNCtion<number>:ONLY command does the same thing as the MAGNIFY command. It defines a function that takes a copy of the original operand. It is useful for scaling channels, another function, and memories with the RANGE and OFFSET commands in this subsystem.

> **The ONLY command is provided for compatibility with previous instruments. MAGNIFY is the preferred method for new programs.**

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example creates a function (function 1) that is a copy of channel 1.

```
10  OUTPUT 707;":FUNCTION1:ONLY CHANNEL1"
20  END
```

RANGe

Command

```
:FUNCtion<number>:RANGe <full_scale_range>
```

The FUNCtion<number>:RANGe command defines the full scale vertical axis of the selected function.

`<number>` An integer, 1 or 2, representing the selected function.

`<full_scale_range>` Full-scale vertical range.

Example

The following example sets the full scale range for function 1 to 400 mV.

```
10  OUTPUT 707;":FUNCTION1:RANGE 400E-3"
20  END
```

Query

```
:FUNCtion<number>:RANGe?
```

The FUNCtion<number>:RANGe query returns the current full scale range setting for the specified function.

Returned Format

```
[:FUNCtion<number>:RANGe] <full_scale_range><NL>
```

Example

The following example places the current range setting for function 2 in the numeric variable "Value," then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

SUBTract

Command

```
:FUNCtion<number>:SUBTract <operand>,<operand>
```

The FUNCtion<number>:SUBTract command defines a function that algebraically subtracts the second operand from the first operand.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example defines a function that subtracts waveform memory 1 from channel 1.

```
10  OUTPUT 707;":FUNCTION1:SUBTRACT CHANNEL1,WMEMORY1"
20  END
```

VERSus

Command

```
:FUNCtion<number>:VERSus <operand>,<operand>
```

The FUNCtion<number>:VERSus command defines a function for an X versus Y display. The first operand defines the y-axis and the second defines the x-axis. The y-axis range and offset is initially equal to that of the first operand and can be adjusted with the RANGE and OFFSET commands in this subsystem.

`<number>` An integer, 1 or 2, representing the selected function.

`<operand>` {CHANnel<N> | FUNCtion1 | WMEMory<N> | <float_value>}

Where <N> is as follows:

For channels: 1, 2, 3, or 4, optionally followed by A or B.

For waveform memories: 1, 2, 3, or 4.

Example

The following example defines function 1 as an x versus y display. Channel 1 is the x-axis and waveform memory 2 is the y-axis.

```
10  OUTPUT 707;":FUNCTION1:VERSUS WMEMORY2,CHANNEL1"
20  END
```

VERTical

Command

```
:FUNCtion<number>:VERTical {TSOurce | MANual | AUTO}
```

The FUNCtion<number>:VERTical command sets the vertical scaling mode of the specified function to either Track SOurce, MANual, or AUTO.

Whether you use TSOurce or AUTO depends on the function definition. TSOurce should be used with the operators MAGNify and VERSus. AUTO should be used with all others. However, the instrument will accept either TSOurce or AUTO for functions that actually use "auto" on the front panel and will put the function into the auto scaling mode.

The VERTical command also contains a subsystem consisting of the following commands and queries:

- OFFset, and
- RANge.

`<number>` An integer, 1 or 2, representing the selected function.

Query

```
:FUNCtion<number>:VERTical?
```

The query returns the current vertical scaling mode of the specified function.

Returned Format

```
[:FUNCtion<number>:VERTical] {TSOurce | MANual | AUTO}<NL>
```

Example

The following example places the current state of the vertical tracking of function 1 in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":FUNCTION1:VERTICAL?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

VERTical:OFFSet

Command

```
:FUNCtion<number>:VERTical:OFFSet <offset_value>
```

The FUNCtion<number>:VERTical:OFFSet command sets the voltage represented at center screen for the selected function.

`<number>` Function number 1 or 2.

`<offset_value>` The offset value is limited only to being within the vertical range that can be represented by the function data.

Query

```
:FUNCtion<number>:VERTical:OFFset?
```

The query returns the current offset value of the selected function.

Returned Format

```
[:FUNCtion<number>:VERTical:OFFset] <offset_value><NL>
```

Example

The following example places the current offset setting for function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:VERTICAL:OFFSET?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VERTical:RANGe

Command

```
:FUNCtion<number>:VERTical:RANGe <full_scale_range>
```

The FUNCtion<number>:VERTical:RANGe command defines the full-scale vertical axis of the selected function.

<number> Function number 1 or 2.

<full_scale_range> The range may be expanded until there are no fewer than 32 q-levels on the screen.

Query

```
:FUNCtion<number>:VERTical:RANGe?
```

The query returns the current range setting of the specified function.

Returned Format

```
[:FUNCtion<number>:VERTical:RANGe] <range><NL>
```

Example

The following example places the current vertical range setting of function 2 in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":FUNCTION2:VERTICAL:RANGe?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

17

Hardcopy Commands

Hardcopy Commands

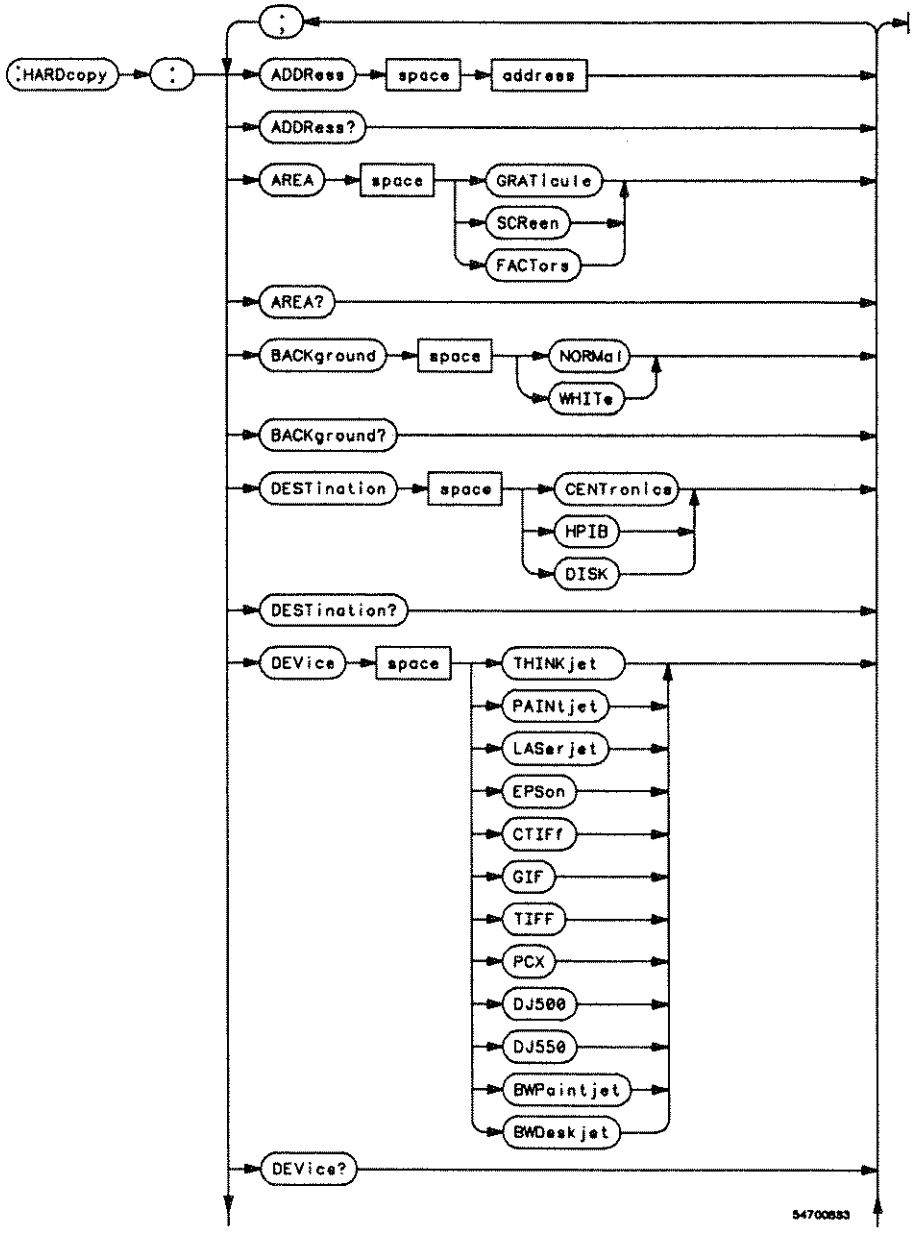
The HARDCOPY subsystem commands set various paramters for printing the screen. The print sequence is activated when the root level command PRINT is sent.

The HARDCOPY subsystem contains the following commands:

- ADDRess,
- AREA,
- BACKground,
- DESTination,
- DEVice,
- FACTors,
- FFEed (Form FEed),
- FILEname,
- LENGth, and
- MEDia.

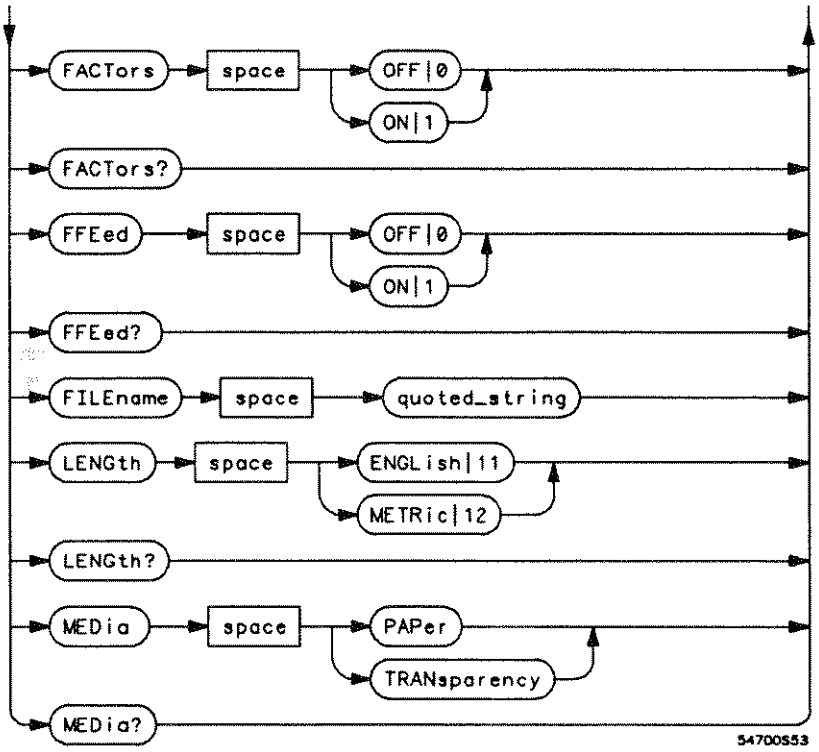
Figure 17-1 is the syntax diagram for the HARDCOPY subsystem commands.

Figure 17-1



Hardcopy Subsystem Syntax Diagram

Figure 17-1 (continued)



Hardcopy Subsystem Syntax Diagram (continued)

ADDRess

Command

```
:HARDcopy:ADDRess <address>
```

The ADDRess command sets the address for the printer when printing to an HP-IB printer.

`<address>` The address must be a legal HP-IB address, 0 to 30.

Example

The following example sets the hardcopy destination to HP-IB address 8.

```
10  OUTPUT 707;":HARDCOPY:DESTINATION HPIB"
20  OUTPUT 707;":HARDCOPY:ADDRESS 8"
30  END
```

Query

```
:HARDcopy:ADDRess?
```

The query returns the current HP-IB printer address.

Returned Format

```
[:HARDcopy:ADDRess] <address>
```

Example

The following example gets the HP-IB address of the hardcopy destination and prints it on the controller screen:

```
10  OUTPUT 707;":HARDCOPY:ADDRESS?"
20  ENTER 707;Hard_address
30  PRINT Hard_address
40  END
```

AREA

Command

```
:HARDcopy:AREA {GRATicule | SCReen | FACTors}
```

The HARDcopy:AREA command selects which data from the screen is to be printed. When GRATicule is selected, the data in the graticule area of the screen is printed. When SCReen is selected, the entire screen is printed. When factors is selected, only the text description of the current setup is printed.

The setup factors can be turned on and off using the HARDcopy:FACTors command when either GRATicule or SCReen is selected as the area for printing.

Example

The following example selects the graticule data for printing.

```
10  OUTPUT 707;":HARDCOPY:AREA GRATICULE"
20  END
```

Query

```
:HARDcopy:AREA?
```

The HARDcopy:AREA query returns the current setting for the area of the screen to be printed.

Returned Format

```
[:HARDcopy:AREA] {GRATicule | SCReen | FACTors}<NL>
```

Example

The following example places the current selection for the area to be printed in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]          !Dimension variable
20  OUTPUT 707;":HARDCOPY:AREA?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

BACKground

Command

```
:HARDcopy:BACKground {WHITe | NORMal}
```

This command controls the background color of the graticule area of an HP PaintJet print. It is only valid when the destination printer is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper).

Example

The following example selects white as the backgound color.

```
10  OUTPUT 707;":HARDCOPY:BACKGROUND WHITE"
20  END
```

Query

```
:HARDcopy:BACKground?
```

The query returns selected setting of the background color.

Returned Format

```
{:HARDcopy:BACKground {WHITe | NORMal}<NL>
```

Example

The following example places the current selection for print destination in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":HARDCOPY BACKGROUND?
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

DESTination

Command

```
:HARDcopy:DESTination {CENTronics | HPIB | DISK}
```

The HARDcopy:DESTination command selects the destination for printing. The options are HPIB, CENTronics, and the internal DISK.

> When printing TIFF, GIF, or PCX files, the only choice of destination is disk.

Example

The following example selects HPIB as the print destination.

```
10  OUTPUT 707;":HARDCOPY:DESTINATION HPIB"
20  END
```

Query

```
:HARDcopy:DESTination?
```

The HARDcopy:DESTination query returns the current destination for printing.

Returned Format

```
[:HARDcopy:DESTination] {CENTronics | HPIB | DISK}<NL>
```

Example

The following example places the current selection for print destination in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":HARDCOPY:DESTINATION?
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

DEVice

Command

```
:HARDcopy:DEVice {THINkjet | PAINtjet | LASerjet |
EPSon | TIFF | CTIFf | GIF | PCX | DJ500 | DJ550 |
BWPaintjet | BWDeskjet}
```

The HARDcopy:DEVice command selects the output format. When TIFF, CTIFf, GIF, or PCX are selected, the only destination available is DISK. To get data in the TIFF, CTIFf, GIF, or PCX format, use the root level PRINt command. The HARDcopy:DESTination DISK command can be used to store the data on disk in any of the above formats.

Example

The following example sets up the output format for an HP LaserJet printer.

```
10  OUTPUT 707;":HARDCOPY:DEVICE LASERJET"
20  END
```

Query

```
:HARDcopy:DEVice?
```

The HARDcopy:DEVice returns the current output format selection.

Returned Format

```
[:HARDcopy:DEVice] {THINkjet | PAINtjet | LASerjet | EPSon |
TIFF | CTIFf | GIF | PCX | DJ500 | DJ550 | BWPaintjet |
BWDeskjet}<NL>
```

Example

The following example places the current selection for the hardcopy output format in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]        !Dimension variable
20  OUTPUT 707;":HARDCOPY:DEVICE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

FACTors

Command

```
:HARDcopy:FACTors {{ON | 1} | {OFF | 0}}
```

The HARDcopy:FACTors command determines whether the instrument setup factors will be appended to screen or graticule images.

Example

The following example turns on the setup factors.

```
10  OUTPUT 707;":HARDCOPY:FACTORS ON"
20  END
```

Query

```
:HARDcopy:FACTors?
```

The HARDcopy:FACTors query returns the current setup factors setting.

Returned Format

```
[:HARDcopy:FACTors] {1 | 0}<NL>
```

Example

The following example places the current setting for the setup factors in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":HARDCOPY:FACTORS?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FFEed (Form FEed)

Command

```
:HARDcopy:FFEed {{ON | 1} | {OFF | 0}}
```

This command sets the form feed option. If it is set to ON, a formfeed will occur at the end of the hardcopy; otherwise, the page will scroll up by 4 lines.

Example

The following example turns the form feed option on.

```
10  OUTPUT 707;":HARDCOPY:FFEED ON"
20  END
```

Query

```
:HARDcopy:FFEed?
```

The HARDcopy:FFEed query returns the current setup of the form feed option setting.

Returned Format

```
[:HARDcopy:FACTors] { 1 | 0}<NL>
```

Example

The following example places the current setting for the setup factors in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":HARDCOPY:FFEED?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

FILEname

Command

```
:HARDcopy:FILEname <string>
```

This command specifies the filename when printing files to the disk.

`<string>` Any series of ASCII characters(a to z, 0 to 9, and, _ underscore) enclosed in quotes and up to 8 characters. The appropriate suffix will be automatically appended.

Example

The following example sets the filename for printing to "CAPTURE."

```
10  OUTPUT 707;":HARDCOPY:FILENAME ""CAPTURE"""
20  END
```

LENGth

Command

```
:HARDcopy:LENGth {{ENGLish | 11} | {METRic | 12}}
```

The HARDcopy:LENGth command sets the length of the paper to either ENGLish (11 inches) or METRic (12 inches). The longer paper is metric size A4.

Example

The following example sets the hardcopy length to 11 inches.

```
10  OUTPUT 707;":HARDCOPY:LENGTH ENGLISH"
20  END
```

Query

```
:HARDcopy:LENGth?
```

The HARDcopy:LENGth query returns the current paper length setting.

Returned Format

```
[:HARDcopy:LENGth] {ENGLish | METRic}<NL>
```

Example

The following example places the current setting for the paper length in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]                !Dimension variable
20  OUTPUT 707;":HARDCOPY:LENGTH?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

MEDia

Command

```
:HARDcopy:MEDia {PAPer | TRANsparency}
```

The HARDcopy:MEDia command sets the speed of the printer. When TRANSPARENCY is selected, the printer prints the data twice, which makes the contents of the paper look darker and slows down the printing process. This command applies only to HP PaintJet and color DeskJet printers.

Example

The following example sets up the printer for printing on paper.

```
10  OUTPUT 707;":HARDCOPY:MEDIA PAPER"
20  END
```

Query

```
:HARDcopy:MEDia?
```

The HARDcopy:MEDia query returns the current media setting.

Returned Format

```
[:HARDcopy:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example places the current setting for printer speed in the string variable, Speed$, then prints the contents of the variable to the controller's screen.

```
10  DIM Speed$[50]                  !Dimension variable
20  OUTPUT 707;":HARDCOPY:MEDIA?"
30  ENTER 707;Speed$
40  PRINT Speed$
50  END
```

18

Marker Commands

Marker Commands

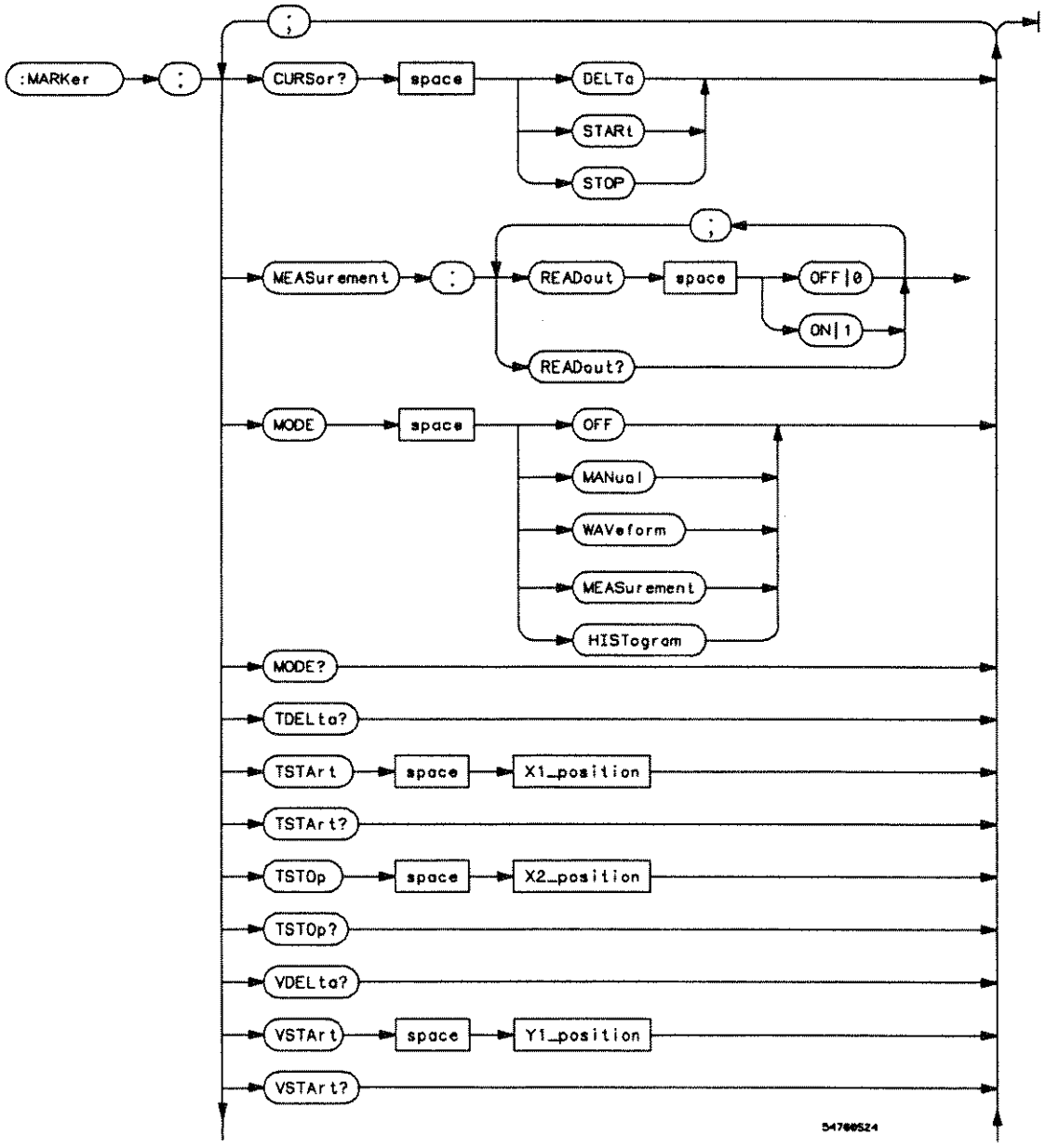
The commands in the MARKER subsystem are used to specify and query the settings of the time (X-axis) and current measurement unit (volts, amps, watts, and so on, for the Y-axis) markers. The Y-axis measurement units are typically set using the CHANnel:UNITs command.

The Marker subsystem contains the following commands and queries:

- CURsor?,
- MEASurement:READout,
- MODE,
- TDELta?,
- TSTArt,
- TSTOp,
- VDELta?,
- VSTArt,
- VSTOp,
- X1Position,
- X2Position,
- X1Y1source,
- X2Y2source,
- XDELta?,
- Y1Position,
- Y2Position, and
- YDELta?.

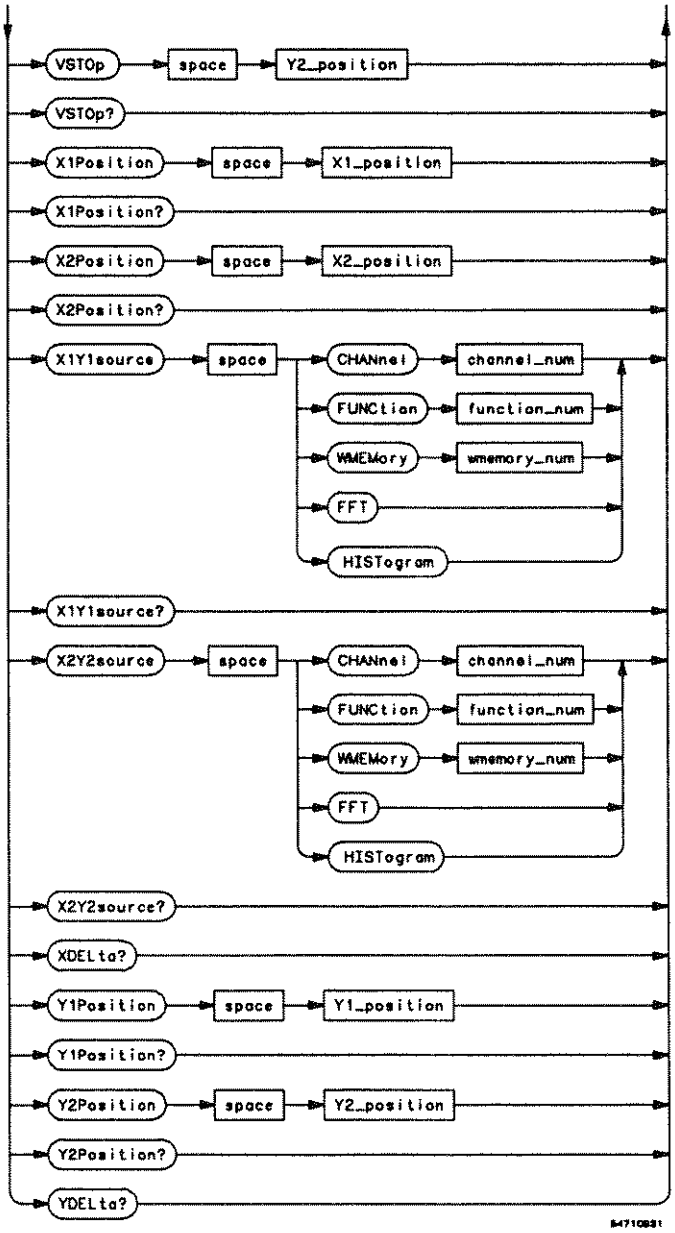
Figure 18-1 lists the syntax diagrams for the Marker subsystem commands.

Figure 18-1



Marker Subsystem Syntax Diagram

Figure 18-1 (continued)



Marker Subsystem Syntax Diagram (continued)

Figure 18-1

X1_position	time value
X2_position	time value
Y1_position	current measurement unit (volts, amps, watts, etc.) value
Y2_position	current measurement unit (volts, amps, watts, etc.) vlaue
channel_num	1, 2, 3, or 4
function_num	1 or 2
wmemory_num	1, 2, 3, or 4

Marker Subsystem Syntax Diagram (continued)

CURSor?

Query

```
:MARKer:CURSor? {DELTa | STARt | STOP}
```

The MARKer:CURSor query returns the time and current measurement unit values of the specified marker or cursor (if in the waveform mode) as an ordered pair of time and measurement unit values.

If DELTA is specified, the value of delta Y and delta X are returned. If START is specified, the position of X1 (or the + cursor) and measurement unit marker 1 (Y1) are returned. If STOP is specified, the positions of X2 (or the X cursor) and measurement unit marker 2 (Y2) are returned.

Returned Format

```
[:MARKer:CURSor] {DELTa | STARt | STOP}
{<X1, Y1> | <X2, Y2> | <deltaX, deltaY>}<NL>
```

Example

The following example returns the current position of the x cursor and measurement unit marker 1 to the string variable, Position$. Then the program prints the contents of the variable to the controller's screen.

```
10  DIM Position$[50]                !Dimension variable
20  OUTPUT 707;":MARKER:CURSOR? START"
30  ENTER 707;Position$
40  PRINT Position$
50  END
```

MEASurement:READout

Command

```
:MARKer:MEASurement:READout {{ON | 1} | {OFF | 0}}
```

The MARker:MEASurement:READout command controls the state of the marker readout area on the screen when markers are on in measurement mode.

Query

```
:MARKer:MEASurement:READout?
```

The query returns the current state of the marker readout display.

Returned Format

```
{:MARKer:MEASurement:READout} {1 | 0}<NL>
```

MODE

Command

```
:MARKer:MODE {OFF | MANual | WAVeform |
MEASurement | HISTogram}
```

The MARKer:MODE command sets the marker mode.

Example

The following example sets the marker mode to waveform.

```
10  OUTPUT 707;":MARKER:MODE WAVEFORM"
20  END
```

Query

```
:MARKer:MODE?
```

The MARKer:MODE query returns the current marker mode.

Returned Format

```
[:MARKer:MODE] {OFF | MANual | WAVeform | MEASurement |
HISTogram}<NL>
```

Example

The following example places the current marker mode in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:MODE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

TDELta?

Query `:MARKer:TDELta?`

The MARKer:TDELta query returns the time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

> The MARKer:TDELta query performs the same function as the MARKer:XDELta query. MARKer:TDELta is provided for compatibility with older instruments. MARKer:XDELta should be used for new programs.

Returned Format `[:MARKer:TDELta] <time><NL>`

`<time>` The time difference between X1 and X2 time markers or between the X cursor and + cursor depending on the current marker mode.

Example

The following example places the time difference between the X1 and X2 markers in the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:TDELTA?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

> When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TSTArt

Command

```
:MARKer:TSTArt <X1 position>
```

The MARKer:TSTArt command sets the X1 position and moves the X1 marker or X cursor, depending on the current marker mode.

> The MARKer:TSTArt command and query perform the same function as the MARKer:X1Position command and query. MARKer:TSTArt is provided for compatibility with previous instruments. MARKer:X1Position should be used for new programs.

`<X1 position>` Time at X1 marker or X cursor in seconds.

Example

The following example sets the X1 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:TSTART 90 NS"
20  END
```

Query

```
:MARKer:TSTArt?
```

The MARKer:TSTArt query returns the time at X1 or X cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:TSTArt] <X1 position><NL>
```

Example

The following example places the current setting of the X cursor in the numeric variable, Setting, then print the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off"
20  OUTPUT 707;":MARKER:TSTART?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

The short form of the TSTArt command and query do not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP. Sending "TST" produces an error.

TSTOp

Command

```
:MARKer:TSTOp <X2 position>
```

The MARKer:TSTOp command sets the X2 position and moves the X2 marker or + cursor, depending on the current marker mode.

> The MARKer:TSTOp command and query perform the same function as the MARKer:X2Position command and query. MARKer:TSTOp is provided for compatibility with previous instruments. MARKer:X2Position should be used for new programs.

`<X2 position>` Time at X2 marker or + cursor in seconds.

Example

The following example sets the X2 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:TSTOP 90 NS"
20  END
```

Query

```
:MARKer:TSTOp?
```

The MARKer:TSTOp query returns the time at X2 or + cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:TSTOp] <X2 position><NL>
```

Example

The following example places the current setting of the + cursor in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off"
20  OUTPUT 707;":MARKER:TSTOP?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

The short form of the TSTOp command and query do not follow the defined convention. The short form "TST" is the same for TSTART and TSTOP. Sending "TST" produces an error.

VDELta?

Query

```
:MARKer:VDELta?
```

The MARKer:VDELta query returns the current measurement unit difference between Y1 and Y2.

> The MARKer:VDELta query performs the same function as the MARKer:YDELta query. MARKer:VDELta is provided for compatibility with previous instruments. MARKer:YDELta should be used for new programs.

Returned Format

```
[:MARKer:VDELta] <value><NL>
```

`<value>` Current measurement unit difference between Y1 and Y2.

Example

The following example returns the voltage difference between Y1 and Y2 to the numeric variable, Volts, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VDELTA?"
30  ENTER 707;Volts
40  PRINT Volts
50  END
```

VSTArt

Command

```
:MARKer:VSTArt <Y1 position>
```

The MARKer:VSTArt command sets the Y1 Position and moves Y1 to the specified measurement unit value on the specified source.

> The MARKer:VSTArt command and query perform the same function as the MARKer:Y1Position command and query. MARKer:VSTArt is provided for compatibility with previous instruments. MARKer:Y1Position should be used for new programs.

`<Y1 position>` Current measurement unit value at Y1.

Example

The following example sets Y1 to −10 mV.

```
10  OUTPUT 707;":MARKER:VSTART -10MV"
20  END
```

Query

```
:MARKer:VSTArt?
```

The MARKer:VSTArt query returns the current measurement unit level of Y1.

Returned Format

```
[:MARKer:VSTArt] <Y1 position><NL>
```

Example

The following example returns the voltage setting for Y1 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VSTART?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP. Sending "VST" produces an error.

VSTOp

Command

```
:MARKer:VSTOp <Y2 position>
```

The MARKer:VSTOp command sets the Y2 position and moves Y2 to the specified measurement unit on the specified source.

> The MARKer:VSTOp command and query perform the same function as the MARKer:Y2Position command and query. MARKer:VSTOp is provided for compatibility with previous instruments. MARKer:Y2Position should be used for new programs.

`<Y2 position>` Current measurement unit value at Y2.

Example

The following example sets Y2 to -100 mV.

```
10  OUTPUT 707;":MARKER:VSTOP -100E-3"
20  END
```

Query

```
:MARKer:VSTOp?
```

The MARKer:VSTOp query returns the current measurement unit level at Y2.

Returned Format

```
[:MARKer:VSTOp] <Y2 position><NL>
```

Example

The following example returns the voltage at Y2 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:VSTOP?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

The short form of this command does not follow the defined convention. The short form "VST" is the same for VSTART and VSTOP. Sending "VST" produces an error.

X1Position

Command

```
:MARKer:X1Position <X1 position>
```

The MARKer:X1Position command sets the X1 position and moves the X1 marker or X cursor, depending on the current marker mode, to the specified time with respect to the trigger time.

`<X1 position>` Time at X1 marker or X cursor in seconds.

Example

The following example sets the X1 marker at 90 ns.

```
10  OUTPUT 707;":MARKER:X1POSITION 90 NS"
20  END
```

Query

```
:MARKer:X1Position?
```

The MARKer:X1Position query returns the time at X1 or the X cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:X1Position] <X1 position><NL>
```

Example

The following example returns the current setting of the X cursor to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:X1POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

MARKer:TSTArt

X2Position

Command

```
:MARKer:X2Position <X2 position>
```

The MARKer:X2Position command sets the X2 position and moves the X2 marker or + cursor, depending on the current marker mode, to the specified time with respect to the trigger time.

`<X2 position>` Time at X2 marker or + cursor in seconds.

Example

The following example sets the X2 marker to 90 ns.

```
10  OUTPUT 707;":MARKER:X2POSITION 90E-9"
20  END
```

Query

```
:MARKer:X2Position?
```

The MARKer:X2Position query returns the time at X2 or the + cursor, depending on the current marker mode.

Returned Format

```
[:MARKer:X2Position] <X2 position><NL>
```

Example

The following example returns the current position of the + cursor to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:X2POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

X1Y1source

Command

```
:MARKer:X1Y1source {CHANnel<number> |
FUNCtion<number> | WMEMory<number>  | HISTogram |
FFT}
```

The MARKer:X1Y1source command sets the source for the X1 and Y1 markers.

<number>

For channels: an integer 1 through 4.

For functions: 1 or 2.

For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the X1Y1 source.

```
10  OUTPUT 707;":MARKER:X1Y1SOURCE CHANNEL1"
20  END
```

Query

```
:MARKer:X1Y1source?
```

The MARKer:X1Y1source query returns the current X1Y1 source.

Returned Format

```
[:MARKer:X1Y1source] {CHANnel<number> | FUNCtion<number> |
WMEMory<number>  | HISTogram | FFT}
```

Example

The following example returns the current X1Y1 source selection to the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:X1Y1SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

X2Y2source

Command

```
:MARKer:X2Y2source {CHANnel<number> |
FUNCtion<number> | WMEMory<number>  | HISTogram|
FFT}
```

The MARKer:X2Y2source command sets the source for the X2 and Y2 markers.

`<number>` For channels: an integer 1 through 4 .

For functions: 1 or 2.

For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the X2Y2 source.

```
10  OUTPUT 707;":MARKER:X2Y2SOURCE CHANNEL1"
20  END
```

Query

```
:MARKer:X2Y2source?
```

The MARKer:X2Y2source query returns the current X2Y2 source.

Returned Format

```
[:MARKer:X2Y2source] {CHANnel<number> | FUNCtion<number> |
WMEMory<number>  | HISTogram| FFT}
```

Example

The following example returns the current X2Y2 source selection to the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]              !Dimension variable
20  OUTPUT 707;":MARKER:X2Y2SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

XDELta?

Query `:MARKer:XDELta?`

The MARKer:XDELta query returns the time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

Manual Mode Xdelta = time at X2 − time at X1

Tracking Mode Xdelta = time at + cursor − time at X cursor

Returned Format `[:MARKer:XDELta] <time><NL>`

`<time>` The time difference between X1 and X2 time markers or between the X cursor and + cursor, depending on the current marker mode.

Example The following example returns the current time between the X1 and X2 time markers to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:XDELTA?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

Y1Position

Command

```
:MARKer:Y1Position <Y1 position>
```

The MARKer:Y1Position command sets the Y1 Position and moves Y1 to the specified measurement unit on the specified source.

`<Y1 position>` Current measurement unit value at Y1.

Example

The following example sets Y1 to 10 mV.

```
10  OUTPUT 707;":MARKER:Y1POSITION 10MV"
20  END
```

Query

```
:MARKer:Y1Position?
```

The MARKer:Y1Position query returns the current measurement unit level of Y1.

Returned Format

```
[:MARKer:Y1Position] <Y1 position>
```

Example

The following example returns the voltage setting for Y1 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"        !Response headers off
20  OUTPUT 707;":MARKER:Y1POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

Y2Position

Command

```
:MARKer:Y2Position <Y2 position>
```

The MARKer:Y2Position command sets the Y2 position and moves Y2 to the specified measurement unit on the specified source.

`<Y2 position>` Current measurement unit value at Y2.

Example

The following example sets Y2 to -100 mV.

```
10  OUTPUT 707;":MARKER:Y2POSITION -100E-3"
20  END
```

Query

```
:MARKer:Y2Position?
```

The MARKer:Y2Position query returns the current measurement unit level at Y2.

Returned Format

```
[:MARKer:Y2Position] <Y2 position><NL>
```

Example

The following example returns the voltage at Y2 to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:Y2POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YDELta?

Query

```
:MARKer:YDELta?
```

The MARKer:YDELta query returns the current measurement unit difference between Y1 and Y2.

Vdelta = value at Y2 − value at Y1

Returned Format

```
[:MARKer:YDELta] <value><NL>
```

<value> Current measurement unit difference between Y1 and Y2.

Example

The following example returns the voltage difference between Y1 and Y2 to the numeric variable, Volts, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MARKER:YDELTA?"
30  ENTER 707;Volts
40  PRINT Volts
50  END
```

19

Measure Commands

Measure Commands

The commands in the MEASURE subsystem are used to make parametric measurements on displayed waveforms.

The Measure subsystem contains the following commands and queries:

- DEFine,
- DELTatime,
- DUTycycle,
- FALLtime,
- FFT:DFREQuency (delta frequency),
- FFT:DMAGnitude (delta magnitude),
- FFT:FREquency,
- FFT:MAGNitude,
- FFT:PEAK1,
- FFT:PEAK2,
- FFT:THReshold,
- FREQuency,
- HISTogram:HITS,
- HISTogram:MEAN,
- HISTogram:MEDian,
- HISTogram:M1S,
- HISTogram:M2S,
- HISTogram:M3S,
- HISTogram:OFFSet?,
- HISTogram:PEAK,
- HISTogram:PP,
- HISTogram:SCALe?,
- NWIDth,
- OVERshoot,
- PERiod,
- PREShoot,
- PWIDth,
- RESults?,
- RISetime,
- SCRatch,
- SENDvalid,
- SOURce,
- STATistics,
- TEDGe,
- TMAX,
- TMIN,
- TVOLt,
- VAMPlitude,
- VAVerage,
- VBASe,
- VLOWer,
- VMAX,
- VMIDdle,
- VMIN,

- HISTogram:STDDev,
- VPP,
- VRMS,
- VTIMe,
- VTOP, and
- VUPPer.

Figure 19-1 is the syntax diagram for the Measure subsystem commands.

Measurement Setup

To make a measurement, the portion of the waveform required for that measurement must be displayed on the oscilloscope.

- For a period or frequency measurement, at least one and one half complete cycles must be displayed.
- For a pulse width measurement, the entire pulse must be displayed.
- For a rise time measurement, the leading (positive-going) edge of the waveform must be displayed.
- For a fall time measurement, the trailing (negative-going) edge of the waveform must be displayed.

User-Defined Measurements

When user-defined measurements are made, the defined parameters must be set before actually sending the measurement command or query.

Measurement Error

If a measurement cannot be made because of the lack of data, because the source signal is not displayed, or some other reason, the following results are returned:

- 9.99999E+37 is returned as the measurement result.

- If SENDVALID is ON, the error code is also returned.

Making Measurements

If more than one period, edge, or pulse is displayed, time measurements are made on the first (Left-most) portion of the displayed waveform.

When any of the defined measurements are requested, the oscilloscope first determines the top (100%) and base (0%) voltages of the waveform. From this information, the oscilloscope determines the other important voltage values (10%, 90%, and 50% voltage values) for making measurements.

The 10% and 90% voltage values are used in the rise-time and fall-time measurements when standard measurements are selected. The 50% voltage value is used for measuring frequency, period, pulse width, and duty cycle with standard measurements selected.

Measurements can also be made using user-defined parameters instead of the standard measurement values.

When the command form of a measurement is used, the oscilloscope is placed in the continuous measurement mode. The measurement result will be displayed on the front panel. There may be up to four measurements running continuously. The SCRATCH command is used to turn off the measurements.

When the query form of the measurement is used, the measurement is made one time, and the measurement result is returned.

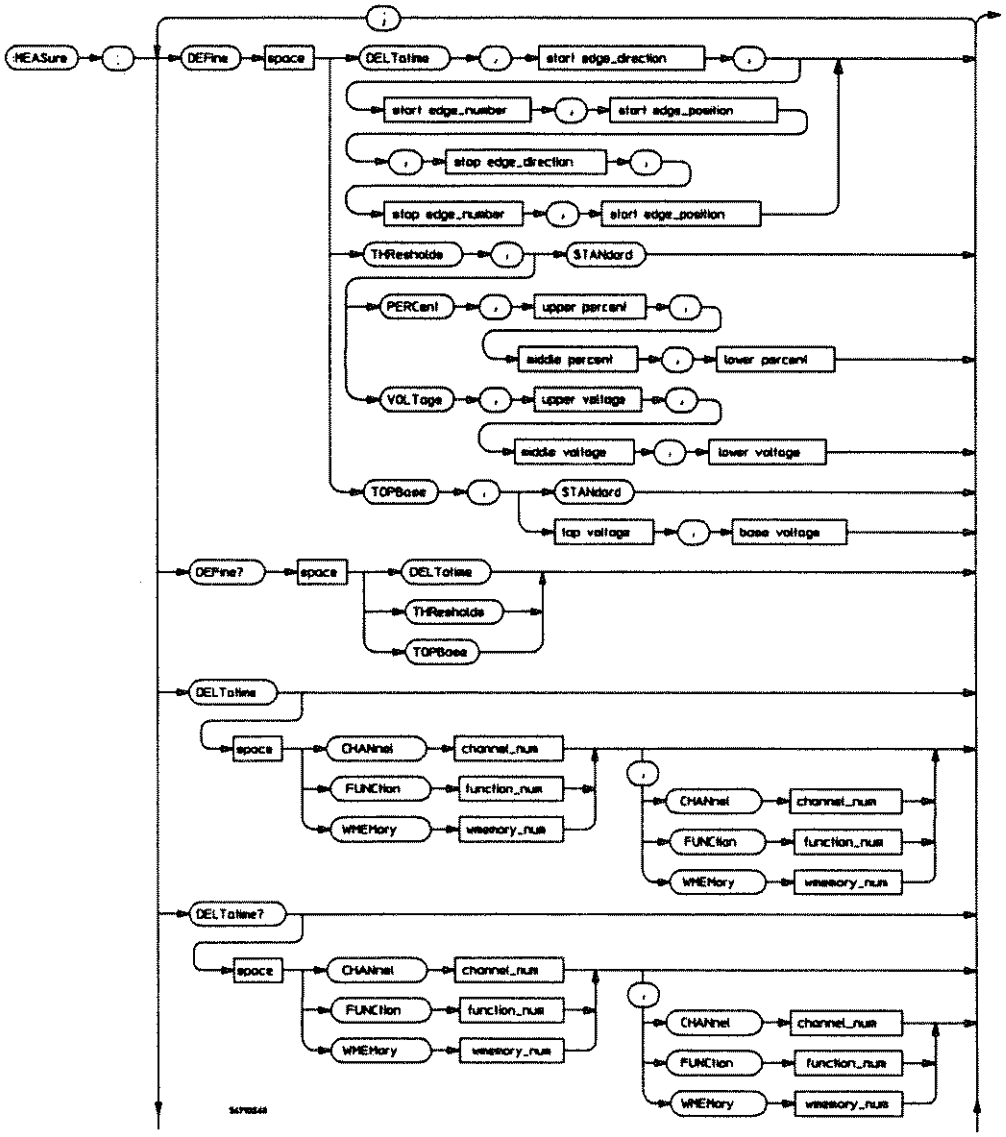
- If the current acquisition is complete, the current acquisition is measured and the result is returned.
- If the current acquisition is incomplete and the instrument is running, acquisitions will continue to occur until the acquisition is complete. The acquisition will then be measured and the result returned.
- If the current acquisition is incomplete and the instrument is stopped, the measurement result will be 9.99999e+37 and the incomplete result state will be returned if SENDVALID is ON.

All measurements are made using the entire display, except for VAVERAGE and VRMS which allow measurements on a single cycle. Therefore, if you want to make measurements on a particular cycle, display only that cycle on the screen.

Measurements are made on the displayed waveforms specified by the SOURCE command. The SOURCE command allows two sources to be specified. Most measurements are only made on a single source. Some measurements, such as the DELTATIME measurement, require two sources.

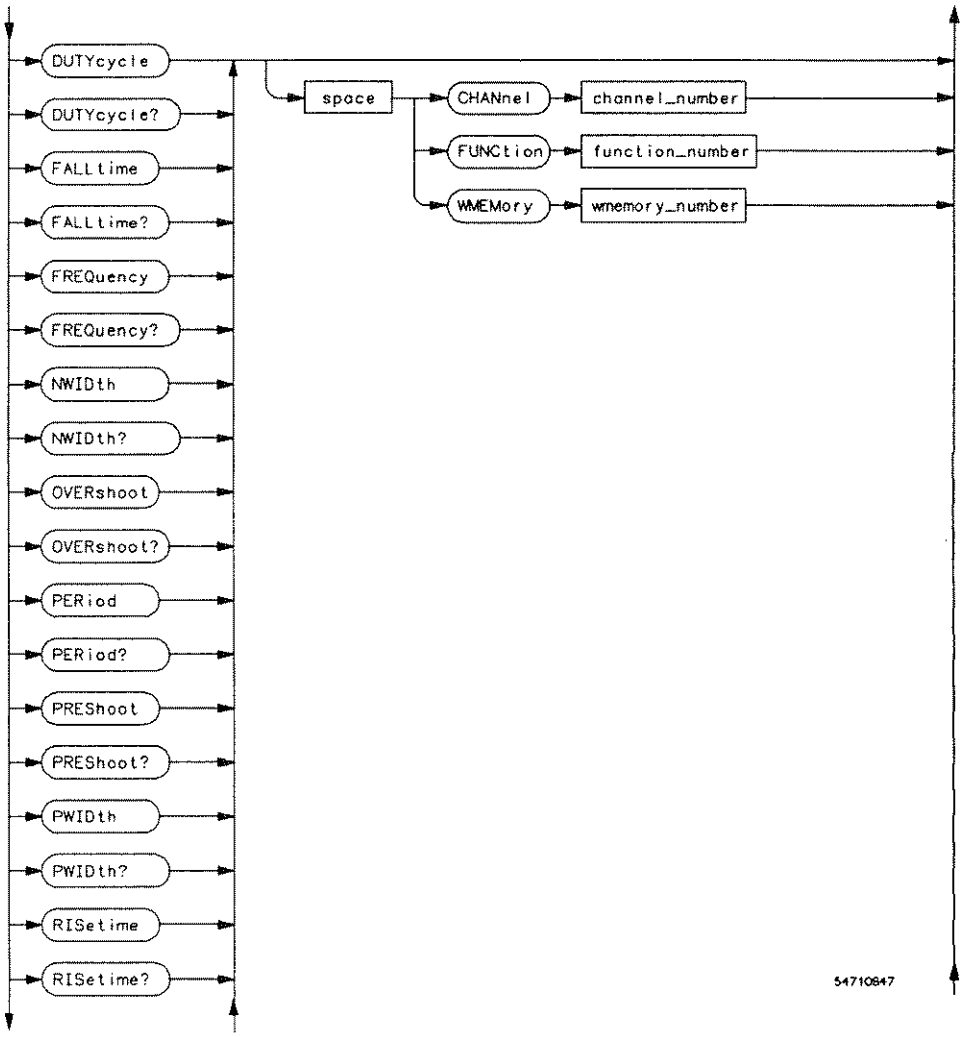
If the signal is clipped, the measurement result may be questionable. In this case, the value returned is the most accurate value that can be made using the current scaling. You might be able to obtain a more accurate measurement by rescaling the vertical to prevent the signal from being clipped.

Figure 19-1



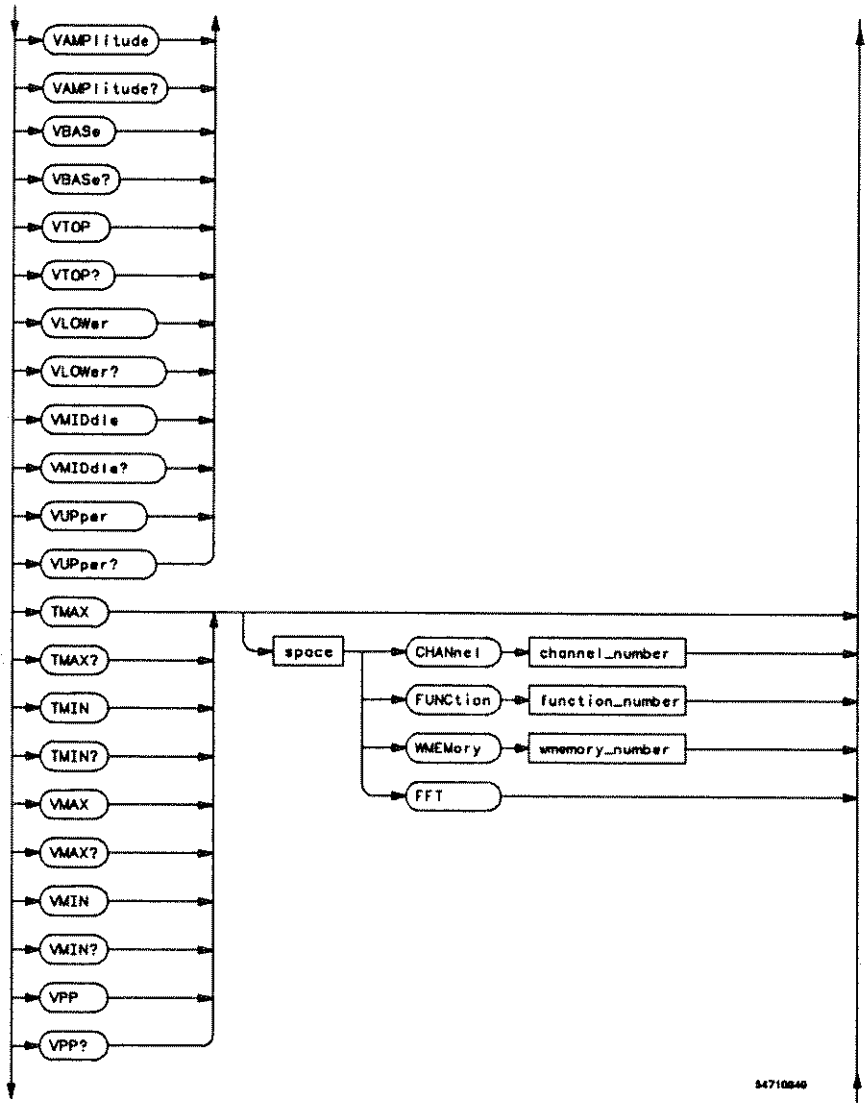
Measure Subsystem Syntax Diagram

Figure 19-1 (continued)



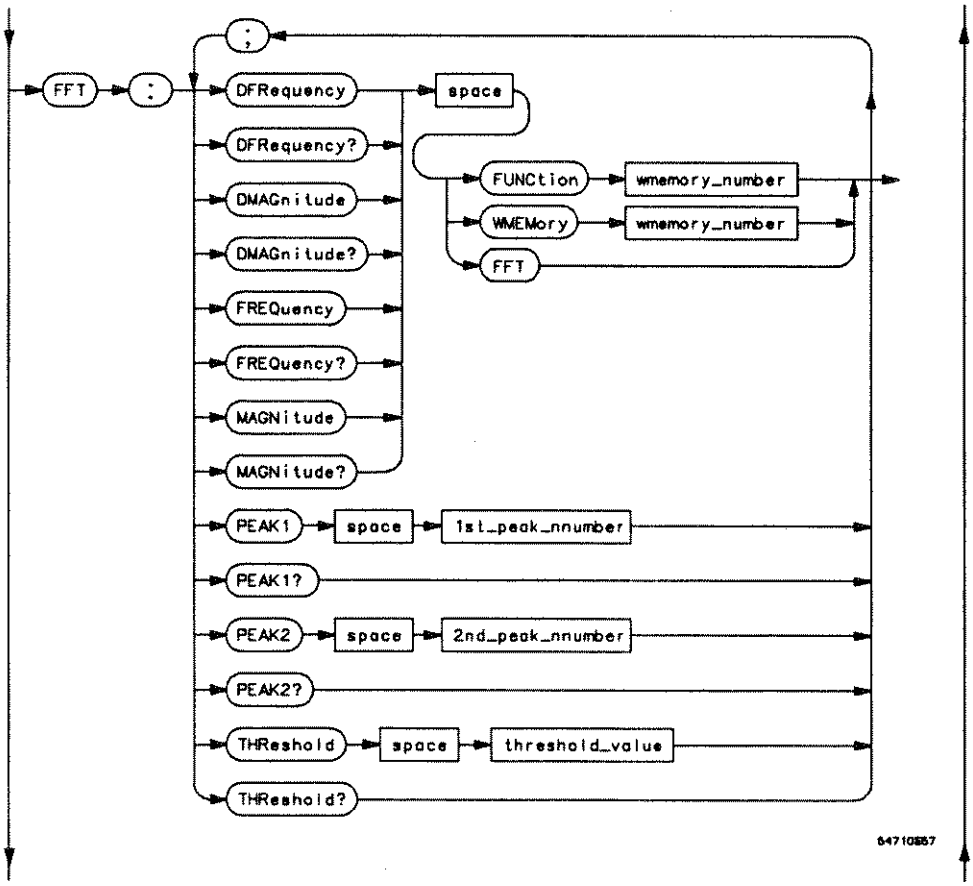
Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)



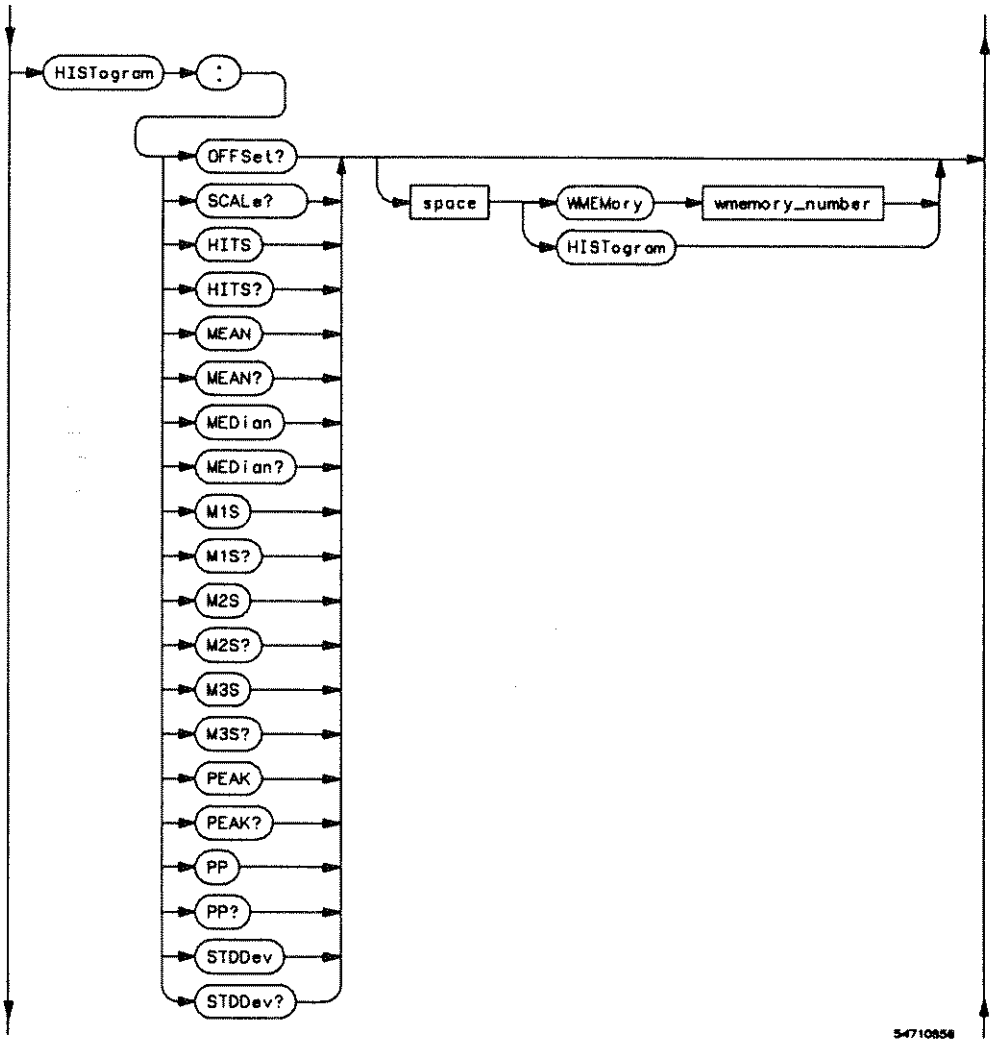
Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)



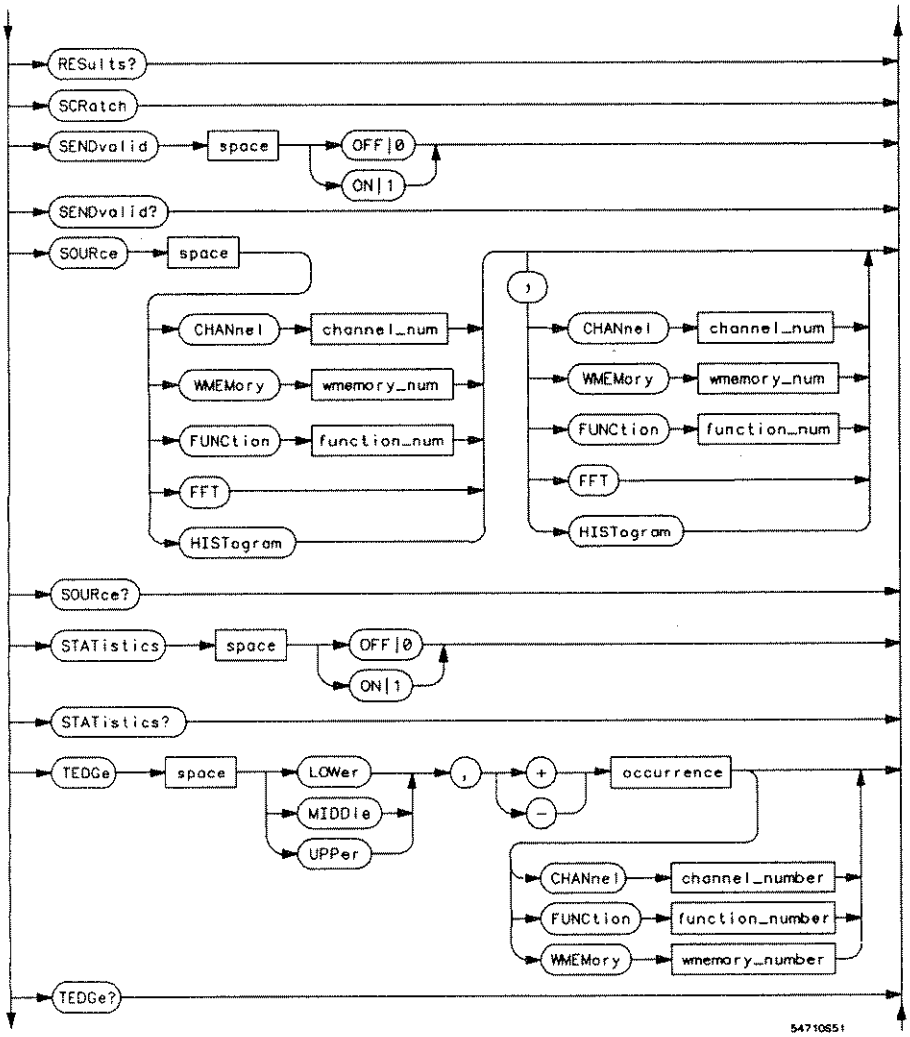
Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)



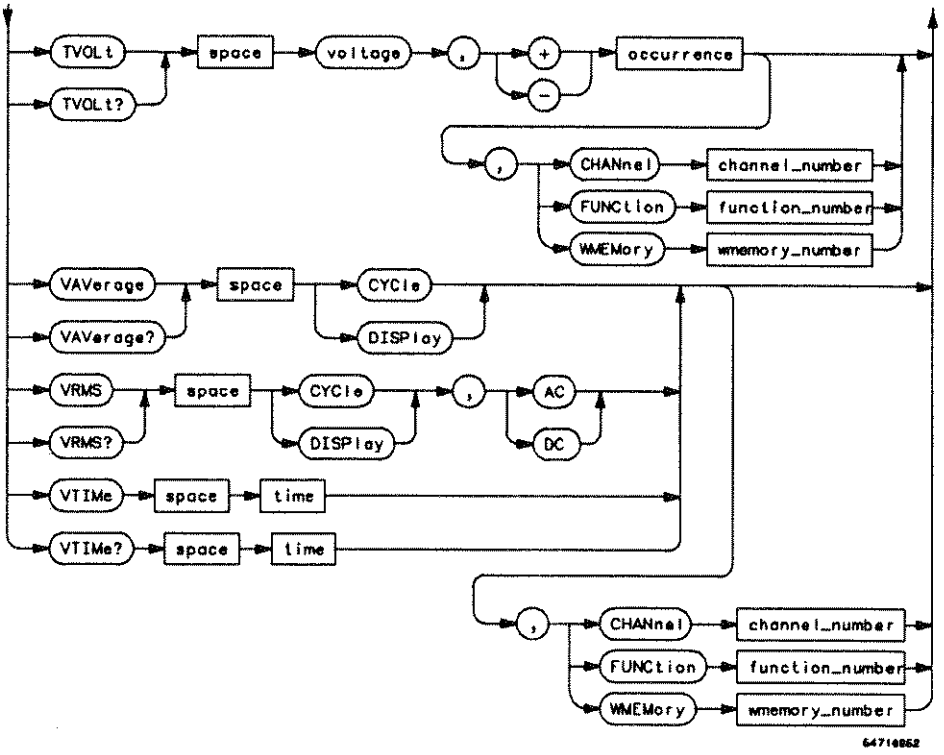
Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)



Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)



Measure Subsystem Syntax Diagram (continued)

Figure 19-1 (continued)

edge_direction	RISing, FALLing, or EITHer.
edge_number	an integer, 1 to 65534.
edge_position	MIDDle, UPPer, or LOWer.
percent	an integer, − 25 to 125.
voltage	a real number specifying voltage.
source	CHANnel, FUNCtion, FFT, or WMEMory number.
channel_num	an integer, 1 to 4.
wmemory_num	an integer, 1 to 4.
function_num	an integer, 1 or 2.
slope	+ or −.
occurrence	an integer, 1 to 65534.
time	a real number specifying time.
1st_peak_number	an integer specifying the number of the first peak
2nd_peak_number	an integer specifying the number of the second peak
threshold_value	a real number specifying the threshold for peaks

Measure Subsystem Syntax Diagram (Continued)

DEFine

Command

```
:MEASure:DEFine <meas_spec>
```

The MEASure:DEFine command sets up the definition for measurements by specifying the delta time, threshold, or top-base values. Changing these values may affect other measure commands. Table 19-1 identifies the relationships between user-DEFined values and other MEASure commands.

`<meas_spec>` `{DELTatime,THResholds,TOPBase}`

Table 19-1 :MEASure:DEFine Interactions

MEASure commands	DELTatime	THResholds	TOPBase
RISEtime		x	x
FALLtime		x	x
PERiod		x	x
FREQuency		x	x
VMIN			
VMAX			
VPP			
VTOP			x
VBASe			x
VAMPlitude			x
PWIDth		x	
NWIDth		x	
OVERshoot		x	
DUTycycle		x	
DELTatime	x	x	
VRMS			

Command
```
:MEASure:DEFine DELTatime,<start edge_direction>,
<start edge_number>,<start edge_position>,
<stop edge_direction>,<stop edge_number>,
<stop edge_position>
```

`<edge_direction>` {RISing | FALLing | EITHer}

`<edge_number>` An integer from 1 to 65534.

`<edge_position>` {UPPer | MIDDle | LOWer}

Command
```
:MEASure:DEFine THResholds,{{STANdard}|
{PERCent,<upper_pct>,<middle_pct>,<lower_pct>} |
{VOLTs,<upper_volts>,<middle_volts>,<lower_volts>}}
```

`<upper_pct>`, `<middle_pct>`, `<lower_pct>` An integer, - 25 to 125.

`<upper_volts>`, `<middle_volts>`, `<lower_volts>` A real number specifying voltage.

Command
```
:MEASure:DEFine TOPBase,{{STANdard}
|{<top_volts>,<base_volts>}}
```

`<top_volts>`, `<base_volts>` A real number specifying voltage.

Example

The following example sets the parameters for a time measurement from the first positive edge at the upper threshold level to the second negative edge at the middle threshold.

```
10  OUTPUT 707;":MEASURE:DEFINE DELTATIME,RISING,
1,UPPER,FALLING,2,MIDDLE"
20  END
```

If one source is specified, both parameters apply to that signal. If two sources are specified, the measurement is from the first positive edge on source 1 to the second negative edge on source 2.

Source is specified either using MEASure:SOURce, or using the optional <source> parameter when the DELTatime measurement is started.

Query

```
:MEASure:DEFine? {DELtatime | THResholds | TOPBase}
```

The MEASure:DEFine query returns the current setup for the specified parameter.

Returned Format

```
[:MEASure:DEFine DELTATIME] <start edge_direction>,
<start edge_number>,<start edge_position>,
<stop edge_direction>,<stop edge_number>,
<stop edge_position>

[:MEASure:DEFine] THRESHOLDS {{STANDARD} |
{PERCENT,<upper_pct>,<middle_pct>,<lower_pct>} |
{VOLTAGE, <upper_volts>,<middle_volts>,<lower_volts>}}

[:MEASure:DEFine] TOPBASE {{STANDARD}
|{<top_volts>,<base_volts>}}
```

Example

The following example returns the current setup for the measurement thresholds to the string variable, Setup$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setup$[50]            !Dimension variable
20  OUTPUT 707;":MEASURE:DEFINE? THRESHOLDS"
30  ENTER 707; Setup$
40  PRINT Setup$
50  END
```

DELTatime

Command

```
:MEASure:DELTatime [<source>[,<source>]]
```

The MEASure:DELTatime command measures the delta time between two edges. If one source is specified, the delta time from the leading edge of the specified source to the trailing edge of the specified source is measured. If two sources are specified, the delta time from the leading edge on the first source to the trailing edge on the second source is measured.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the MEASure:DELTatime command. The rest of the parameters for this command are specified with the MEASure:DEFine command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the delta time between channel 1 and channel 2.

```
10  OUTPUT 707;":MEASURE:DELTATIME CHANNEL1,CHANNEL2"
20  END
```

Query

```
:MEASure:DELTatime? [<source>[,<source>]]
```

The MEASure:DELTatime query returns the measured delta time value.

Returned Format

```
[:MEASure:DELTatime] <value>[,<result_state>]<NL>
```

`<value>` Delta time from the first specified edge on one source to the next specified edge on another source.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value of delta time in the numeric variable, Value, then prints the contents of the variable to the controller's screen. This example assumes the source was set using MEASure:SOURce.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:DELTATIME?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

DUTycycle

Command

```
:MEASure:DUTycycle [<source>]
```

The MEASure:DUTycycle command measures the ratio of the positive pulse width to the period. Sources are specified with the MEASure:SOURce command or with the optional parameter following the DUTycycle command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.
For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the duty cycle of the last specified signal.

```
10  OUTPUT 707;":MEASURE:DUTYCYCLE"
20  END
```

Query `:MEASure:DUTycycle? [<source>]`

The MEASure:DUTycycle query returns the measured duty cycle of the specified source.

Returned Format `[:MEASure:DUTYcycle] <value>[,<result_state>]<NL>`

`<value>` The ratio of the positive pulse width to the period.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example The following example places the current dutycycle of the specified signal in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:DUTYCYCLE?
30  ENTER 707;Value
40  PRINT Value
50  END
```

FALLtime

Command

```
:MEASure:FALLtime [<source>]
```

The MEASure:FALLtime command measures the time at the upper threshold of the falling edge, measures the time at the lower threshold of the falling edge, then calculates the fall time. Sources are specified with the MEASure:SOURce command or with the optional parameter following the FALLtime command.

The first displayed falling edge is used for the fall time measurement. Therefore, for best measurement accuracy, set the sweep speed as fast as possible while leaving the falling edge of the waveform on the display.

Fall time = time at lower threshold point − time at upper threshold point.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the fall time of the last specified signal.

```
10  OUTPUT 707;":MEASURE:FALLTIME"
20  END
```

Query

```
:MEASure:FALLtime?[<source>]
```

The MEASure:FALLtime query returns the fall time of the specified source.

Returned Format

```
[:MEASure:FALLtime] <value>[,<result_state>]<NL>
```

`<value>` Time at lower threshold - time at upper threshold.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value for fall time in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:FALLTIME?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

FFT

The :MEASure:FFT commands control the FFT measurements accessible through the Measure subsystem. The FFT sub-subsystem consists of the following commands:

- DFRequency (delta frequency),
- DMAGnitude (delta magnitude),
- FREQuency,
- MAGNitude,
- PEAK1,
- PEAK2, and
- THReshold.

FFT:DFRequency

Command `:MEASure:FFT:DFRequency [<source>]`

This command enables the delta frequency measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

`<source>` {FUNCtion<number> | FFT | WMEMory<number>}

`<number>` For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

Query `:MEASure:FFT:DFRequency? [<source>]`

The query returns the FFT delta frequency of the specified peaks.

Returned Format `[:MEASure:FFT:DFRequency] <delta_frequency><NL>`

FFT:DMAGnitude

Command `:MEASure:FFT:DMAGnitude [<source>]`

This command enables the delta magnitude measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

`<source>` {FUNCtion<number> | FFT | WMEMory<number>}

`<number>` For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

Query `:MEASure:FFT:DMAGnitude? [<source>]`

The query returns the delta magnitude of the specified peaks.

Returned Format `[:MEASure:FFT:DMAGnitude] <delta_magnitude><NL>`

FFT:FREQuency

Command `:MEASure:FFT:FREQuency [<source>]`

This command enables the frequency measurement. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

`<source>` {FUNCtion<number> | FFT | WMEMory<number>}

`<number>` For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

Query `:MEASure:FFT:FREQuency? [<source>]`

The query returns the frequency measurement.

Returned Format `[:MEASure:FFT:FREQuency] <frequency><NL>`

FFT:MAGNitude

Command

```
:MEASure:FFT:MAGNitude [<source>]
```

This command measures the magnitude of the FFT. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

`<source>` {FUNCtion<number> | FFT | WMEMory<number>}

`<number>` For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:MAGNitude
```

The query returns the magnitude value of the FFT.

Returned Format

```
[:MEASure:FFT:FMAGNitude] <magnitude><NL>
```

FFT:PEAK1

Command

```
:MEASure:FFT:PEAK1 <1st_peak_number> [<source>]
```

`<1st_peak_number>` An integer specifying the number of the first peak.

This command sets the peak number of the first peak for FFT measurements. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

`<source>` {FUNCtion<number> | FFT | WMEMory<number>}

`<number>` For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

Query

```
:MEASure:FFT:PEAK1? [<source>]
```

The query returns the peak number currently set as the first peak.

Returned Format

```
[:MEASure:FFT:PEAK1] <1st_peak_number><NL>
```

FFT:PEAK2

Command

```
:MEASure:FFT:PEAK2 <2nd_peak_number> [<source>]
```

This command sets the peak number of the second peak for FFT measurements. The source is specified with the MEASure:SOURce command or with the optional parameter following the FFT command.

<source> {FUNCtion<number> | FFT | WMEMory<number>}

<number> For functions: 1 or 2.
For waveform memories: 1, 2, 3, or 4.

<2nd_peak_number> An integer specifying the number of the second peak.

Query

```
:MEASure:FFT:PEAK2? [<source>]
```

The query returns the peak number currently set as the second peak.

Returned Format

```
[:MEASure:FFT:PEAK1] <2nd_peak_number><NL>
```

FFT:THReshold

Command

```
:MEASure:FFT:THReshold  <threshold_value>
```

This command sets the peak search threshold value.

Query

```
:MEASure:FFT:THReshold?
```

The query returns the peak search threshold value.

Returned Format

```
[:MEASure:FFT:THReshold] <threshold_value><NL>
```

The following :MEASure commands also operate on FFT functions:

Measure Command	Measurement Performed
:TMAX	The frequency of the maximum value in the spectrum.
:TMIN	The frequency of the minimum value in the spectrum.
:VMAX	The maximum value in the spectrum.
:VMIN	The minimum value in the spectrum.
:VPP	The range of values in the spectrum.
:VTIM	The value at a specified frequency.

FREQuency

Command

`:MEASure:FREQuency [<source>]`

The MEASure:FREQuency command measures the frequency of the first complete cycle on the screen using the mid threshold levels of the waveform (50% levels if standard measurements are selected). The source is specified with the MEASure:SOURce command or with the optional parameter following the FREQuency command.

The algorithm is:

If the first edge on screen is rising then

frequency = 1/(time at second rising edge - time at first rising edge)

else

frequency = 1/(time at second falling edge - time at first falling edge).

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the frequency of the last specified signal.

```
10  OUTPUT 707;":MEASURE:FREQUENCY"
20  END
```

Query

```
:MEASure:FREQuency? [<source>]
```

The MEASure:FREQuency query returns the measured frequency.

Returned Format

```
[:MEASure:FREQuency] <value>[,<result_state>]<NL>
```

<value> The frequency value in Hertz of the first complete cycle on the screen using the mid-threshold levels of the waveform.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current frequency of the signal in the numeric variable, Freq, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:FREQUENCY?"
30  ENTER 707;Freq
40  PRINT Freq
50  END
```

HISTogram:HITS

Command

```
:MEASURE:HISTogram:HITS [<source>]
```

The HISTogram:HITS command measures the number of hits within the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the HITS command. The HISTogram:HITS measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the number of hits within the histogram stored in WMEMory1.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:HITS WMEMORY1"
20  END
```

Query

```
:MEASure:HISTogram:HITS? [<source>]
```

The HISTogram:HITS query returns the number of hits within the histogram.

Returned Format

```
[:MEASure:HISTogram:HITS] <value>[,<result_state>]<NL>
```

`<value>` The number of hits in the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the number of hits within the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:HITS?"
30  ENTER 707;Histhits
40  PRINT Histhits
50  END
```

HISTogram:MEAN

Command

```
:MEASURE:HISTogram:MEAN [<source>]
```

The HISTogram:MEAN command measures the mean of the histogram. The mean of the histogram is the average value of all the points in the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the MEAN command. The HISTogram:MEAN measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the mean of the histogram source.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:MEAN HISTOGRAM"
20  END
```

Query

```
:MEASure:HISTogram:MEAN? [<source>]
```

The HISTogram:MEAN query returns the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:MEAN] <value>[,<result_state>]<NL>
```

`<value>` The mean of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:MEAN?"
30  ENTER 707;Histmean
40  PRINT Histmean
50  END
```

HISTogram:MEDian

Command

```
:MEASURE:HISTogram:MEDian [<source>]
```

The HISTogram:MEDian command measures the median of the histogram. The median of the histogram is the time or voltage of the point at which 50% of the histogram is to the left or right (above or below for vertical histograms).

Sources are specified with the MEASure:SOURce command or with the optional parameter following the MEDian command. The HISTogram:MEDian measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the median of the histogram whose source has been defined with the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:MEDIAN"
20  END
```

Query

```
:MEASure:HISTogram:MEDian? [<source>]
```

The HISTogram:MEDian query returns the median of the histogram.

Returned Format

```
[:MEASure:HISTogram:MEDian] <value>[,<result_state>]<NL>
```

`<value>` The median of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the median of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:MEDIAN?"
30  ENTER 707;Histmed
40  PRINT Histmed
50  END
```

HISTogram:M1S

Command

```
:MEASURE:HISTogram:M1S [<source>]
```

The HISTogram:M1S command measures the percentage of points that are within one standard deviation of the mean of the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the M1S command. The HISTogram:M1S measurement only applies to the histogram waveform or memories containing histograms.

<source> { WMEMory<number> | HISTogram}

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the percentage of points that are within one standard deviation of the mean of the histogram of the data stored in waveform memory 3.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M1S WMEMORY3"
20  END
```

Query `:MEASure:HISTogram:M1S? [<source>]`

The HISTogram:M1S query returns the percentage of points within one standard deviation of the mean of the histogram.

Returned Format `[:MEASure:HISTogram:M1S] <value>[,<result_state>]<NL>`

`<value>` The mean plus/minus one standard deviation of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the percentage of points within one standard deviation of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M1S?"
30  ENTER 707;Histm1s
40  PRINT Histm1s
50  END
```

HISTogram:M2S

Command

```
:MEASURE:HISTogram:M2S [<source>]
```

The HISTogram:M2S command measures the percentage of points that are within two standard deviations of the mean of the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the M2S command. The HISTogram:M2S measurement only applies to the histogram waveform or memories containing histograms.

<source> { WMEMory<number> | HISTogram}

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the percentage of points within two standard deviations of the mean of the histogram whose source is specified using the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M2S"
20  END
```

Query

```
:MEASure:HISTogram:M2S? [<source>]
```

The HISTogram:M2S query returns the percentage of points within two standard deviations of the mean of the histogram.

Returned Format

```
[:MEASure:HISTogram:M2S] <value>[,<result_state>]<NL>
```

`<value>` The mean plus/minus two standard deviations of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the percentage of points within two standard deviations of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M2S?"
30  ENTER 707;Histm2s
40  PRINT Histm2s
50  END
```

HISTogram:M3S

Command

```
:MEASURE:HISTogram:M3S [<source>]
```

The HISTogram:M3S command measures the percentage of points that are within three standard deviations of the mean of the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the M3S command. The HISTogram:M3S measurement only applies to the histogram waveform or memories containing histograms.

<source> {WMEMory<number> | HISTogram}

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the percentage of points that are within three standard deviations of the mean of the histogram.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:M3S HISTOGRAM"
20  END
```

Query `:MEASure:HISTogram:M3S? [<source>]`

The HISTogram:M3S query returns the percentage of points within three standard deviations of the mean of the histogram.

Returned Format `[:MEASure:HISTogram:M3S] <value>[,<result_state>]<NL>`

`<value>` The mean plus/minus three standard deviations of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the percentage of points within three standard deviations of the mean of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:M3S?"
30  ENTER 707;Histm3s
40  PRINT Histm3s
50  END
```

HISTogram:OFFSet?

Query

```
:MEASURE:HISTogram:OFFSet? [<source>]
```

The HISTogram:OFFSet query returns the offset of the histogram in hits or decibels. The offset is in hits for linear-scaled histograms and in decibels for logarithmically-scaled histograms.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the OFFSet query. The HISTogram:OFFSet measurement only applies to the histogram waveform or memories containing histograms.

<source> { WMEMory<number> | HISTogram}

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:HISTogram:OFFSet] <value>[,<result_state>]<NL>
```

<value> The offset of the histogram in hits.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the offset of the current linear-scaled histogram in hits and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:OFFSET?"
30  ENTER 707;Histoff
40  PRINT Histoff
50  END
```

HISTogram:PEAK

Command

```
:MEASURE:HISTogram:PEAK [<source>]
```

The HISTogram:PEAK command measures the number of hits in the greatest peak of the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the PEAK command. The HISTogram:PEAK measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the number of hits in the greatest peak of the histogram whose source is waveform memory 2.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:PEAK WMEMORY2"
20  END
```

Query `:MEASure:HISTogram:PEAK? [<source>]`

The HISTogram:PEAK query returns the number of hits in the greatest peak of the histogram.

Returned Format `[:MEASure:HISTogram:PEAK] <value>[,<result_state>]<NL>`

<value> The number of hits in the histogram peak.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the number of hits in the greatest peak of the current histogram and prints the result to the contoller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:PEAK?"
30  ENTER 707;Histpeak
40  PRINT Histpeak
50  END
```

HISTogram:PP

Command

```
:MEASURE:HISTogram:PP [<source>]
```

The HISTogram:PP command measures the width of the histogram. The width is measured as the time or voltage of the last histogram bucket with data in it minus the time or voltage of the first histogram bucket with data in it.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the PP command. The HISTogram:PP measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the width of the histogram.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:PP HISTOGRAM"
20  END
```

Query

```
:MEASure:HISTogram:PP? [<source>]
```

The HISTogram:PP query returns the width of the histogram.

Returned Format

```
[:MEASure:HISTogram:PP] <value>[,<result_state>]<NL>
```

`<value>` The width of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the width of the current histogram and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:PP?"
30  ENTER 707;Histpp
40  PRINT Histpp
50  END
```

HISTogram:SCALe?

Query

```
:MEASURE:HISTogram:SCALe? [<source>]
```

The HISTogram:SCALe query returns the scale of the histogram in hits or decibels per division. The scale is in hits for linear-scaled histograms and in decibels per division for logarithmically-scaled histograms.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the SCALe query. The HISTogram:SCALe measurement only applies to the histogram waveform or memories containing histograms.

<source> { WMEMory<number> | HISTogram}

<number> For waveform memories (WMEMory): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:HISTogram:SCALe] <value>[,<result_state>]<NL>
```

<value> The scale of the histogram in hits.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the scale of the histogram whose source is specified in MEASure:SOURce and prints the result to the controller's scren.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:SCALE?"
30  ENTER 707;Histscal
40  PRINT Histscal
50  END
```

HISTogram:STDDev

Command

```
:MEASURE:HISTogram:STDDev [<source>]
```

The HISTogram:STDDev command measures the standard deviation of the histogram.

Sources are specified with the MEASure:SOURce command or with the optional parameter following the STDDev command. The HISTogram:STDDev measurement only applies to the histogram waveform or memories containing histograms.

`<source>` { WMEMory<number> | HISTogram}

`<number>` For waveform memories (WMEMory): 1, 2, 3, or 4.

Example

The following example measures the standard deviation of the histogram whose source is specified using the MEASure:SOURce command.

```
10  OUTPUT 707;":MEASURE:HISTOGRAM:STDDEV"
20  END
```

Query `:MEASure:HISTogram:STDDev? [<source>]`

The HISTogram:STDDev query returns the standard deviation of the histogram.

Returned Format `[:MEASure:HISTogram:STDDev] <value>[,<result_state>]<NL>`

`<value>` The standard deviation of the histogram.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the standard deviation of the histogram whose source is specified using the MEASure:SOURce command, and prints the result to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:HISTOGRAM:STDDEV?"
30  ENTER 707;Histstdd
40  PRINT Histstdd
50  END
```

NWIDth

Command

```
:MEASure:NWIDth [<source>]
```

The MEASure:NWIDth command measures the width of the first negative pulse on the screen using the mid threshold levels of the waveform (50% levels with standard measurements selected). Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the NWIDTH command.

The algorithm is:

If the first edge on screen is rising then

nwidth = time at the second rising edge − time at the first falling edge

else

nwidth = time at the first rising edge − time at the first falling edge.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the width of the first negative pulse on screen.

```
10  OUTPUT 707;":MEASURE:NWIDTH"
20  END
```

Query

```
:MEASure:NWIDth? [<source>]
```

The MEASure:NWIDth query returns the measured width of the first negative pulse of the specified source.

Returned Format

```
[:MEASure:NWIDth] <value>[,<result_state>]<NL>
```

`<value>` The width of the first negative pulse on the screen using the mid-threshold levels of the waveform.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current width of the first negative pulse on screen in the numeric variable, Width, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:NWIDTH?"
30  ENTER 707;Width
40  PRINT Width
50  END
```

OVERshoot

Command

```
:MEASure:OVERshoot [<source>]
```

The MEASure:OVERshoot command measures the overshoot of the first edge on the screen. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the OVERSHOOT command.

The algorithm is:

If the first edge on screen is rising, then

overshoot = (Local Vmax − Vtop) / Vamplitude

else

overshoot = (Vbase − Local Vmin) / Vamplitude.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the overshoot of the first edge on screen.

```
10  OUTPUT 707;":MEASURE:OVERSHOOT"
20  END
```

Query

```
:MEASure:OVERshoot? [<source>]
```

The MEASure:OVERshoot query returns the measured overshoot of the specified source.

Returned Format

```
[:MEASure:OVERshoot] <value>[,<result_state>]<NL>
```

<value> Ratio of overshoot to amplitude, in percent.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value of overshoot in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:OVERSHOOT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

PERiod

Command

```
:MEASure:PERiod [<source>]
```

The MEASure:PERiod command measures the period of the first complete cycle on the screen using the mid threshold levels of the waveform (50% levels with standard measurements selected). Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the PERIOD command.

The algorithm is:

If the first edge on screen is rising then

period = time at the second rising edge − time at the first rising edge

else

period = time at the second falling edge − time at the first falling edge.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the period of the waveform.

```
10  OUTPUT 707;":MEASURE:PERIOD"
20  END
```

Query `:MEASure:PERiod? [<source>]`

The MEASure:PERiod query returns the measured period of the specified source.

Returned Format `[:MEASure:PERiod] <value>[,<result_state>]<NL>`

`<value>` Period of the first complete cycle on screen.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current period of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PERIOD?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

PREShoot

Command

```
:MEASure:PREShoot [<source>]
```

The MEASure:PREShoot command measures the preshoot of the first edge on the screen. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the PRESHOOT command.

The algorithm is:

If the first edge on screen is rising then

preshoot = (Vbase − Local Vmin) / Vamplitude

else

preshoot = (Local Vmax − Vtop) / Vamplitude.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the preshoot of the signal on screen.

```
10  OUTPUT 707;":MEASURE:PRESHOOT"
20  END
```

Query

```
:MEASure:PREShoot?[<source>]
```

The MEASure:PREShoot query returns the measured preshoot of the specified source.

Returned Format

```
[:MEASure:PREShoot] <value>[,<result state>]<NL>
```

`<value>` Ratio of preshoot to amplitude, in percent.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value of preshoot in the numeric variable, Preshoot, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PRESHOOT?"
30  ENTER 707;Preshoot
40  PRINT Preshoot
50  END
```

PWIDth

Command

```
:MEASure:PWIDth [<source>]
```

The MEASure:PWIDth command measures the width of the first positive pulse on the screen using the mid threshold levels of the waveform (50% levels with standard measurements selected). Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the PWIDTH command.

The algorithm is:

If the first edge on screen is rising then

pwidth = time at the first falling edge − time at the first rising edge

else

pwidth = time at the second falling edge − time at the first rising edge.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the width of the first positive pulse on the screen.

```
10  OUTPUT 707;":MEASURE:PWIDTH"
20  END
```

Query `:MEASure:PWIDth?[<source>]`

The MEASure:PWIDth query returns the measured width of the first positive pulse of the specified source.

Returned Format `[:MEASure:PWIDth] <value>[,<result_state>]<NL>`

`<value>` Width of the first positive pulse on the screen in seconds.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the value of the width of the first positive pulse on the screen in the numeric variable, Width, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:PWIDTH?"
30  ENTER 707;Width
40  PRINT Width
50  END
```

RESults?

Query

```
:MEASure:RESults?
```

The MEASure:RESults query returns the results of the continuous measurements. The measurement results always include only the current results. If SENDVALID is ON, then the measurement results state is returned immediately following the measurement result. If statistics are computed, the measurement results include the current, minimum, maximum, mean, standard deviation, and statistical sample size of each measurement. If the limittest is on, then limittest fields of failures, total waveforms, and status are returned. Table 19-2 shows the relationship of the values returned and STATISTICS, SENDVALID, and LIMITTEST commands.

If more than one measurement is running continuously, then the values in table 19-3 will be duplicated for each continuous measurement from the first to last (top to bottom) of display. There may be up to four continuous measurements at a time.

Table 19-2

Results Values Returned

	SENDVALID OFF	SENDVALID OFF
LIMITTEST OFF **STATISTISC OFF**	current	current, result state
LIMITTEST OFF **STATISTICS ON**	current, minimum, maximum, standard deviation, sample size	current, result state, minimum, maximum, mean, standard deviation, sample size
LIMITTEST ON **STATISTISC OFF** **LTEST:SSUM:FORM:BRIEF**	current, failures, total, status	current, result state, failures, total, status
LIMITTEST ON **STATISTISC OFF** **LTEST:SSUM:FORM:BRIEF**	current, minimum, maximum, mean, standard deviation, sample size, failures, total, status	current, result state, minimum, maximum, mean, standard deviation, sample size, failures, total, status

Returned Format

```
[:MEASure:RESults] <result list><NL>
```

`<result list>` List of the measurement results, as listed in table 19-3, separated with commas.

Example

The following example places the current results of the measurements in the string variable, Result$, then prints the contents of the variable to the controller's screen.

```
10  DIM Result$[200]          !Dimension variable
20  OUTPUT 707;":MEASURE:RESULTS?"
30  ENTER 707;Result$
40  PRINT Result$
50  END
```

Table 19-3 **Result States**

Result State Code	Result	Description
0	RESULT_CORRECT	Result correct. No problem found.
1	RESULT_QUESTIONABLE	Result questionable but could be measured.
2	RESULT_LESS_EQ	Result less than or equal to value returned.
3	RESULT_GTR_EQ	Result greater than or equal to value returned.
4	RESULT_INVALID	Result returned is invalid.
5	EDGE_NOT_FOUND	Result invalid. Required edge not found.
6	MAX_NOT_FOUND	Result invalid. Max not found.
7	MIN_NOT_FOUND	Result invalid. Min not found.
8	TIME_NOT_FOUND	Result invalid. Requested time not found.
9	VOLT_NOT_FOUND	Result invalid. Requested voltage not found.
10	TOP_EQUALS_BASE	Result invalid. Top and base are equal.
11	MEAS_ZONE_SMALL	Result invalid. Measurement zone too small.
12	LOWER_INVALID	Result invalid. Lower threshold not on waveform.
13	UPPER_INVALID	Result invalid. Upper threshold not on waveform.
14	UPPER_LOWER_INVALID	Result invalid. Upper and lower thresholds are too close.
15	TOP_INVALID	Result invalid. Top not on waveform.
16	BASE_INVALID	Result invalid. Base not on waveform.
17	INCOMPLETE	Result invalid. Completion criteria not reached.

18	INVALID_SIGNAL	Result invalid. Measurement invalid for this type of signal.
19	SIGNAL_NOT_DISPLAYED	Result invalid. Signal is not displayed.
20	CLIPPED_HIGH	Result invalid. Waveform is clipped high.
21	CLIPPED_LOW	Result invalid. Waveform is clipped low.
22	CLIPPED_HIGH_LOW	Result invalid. Waveform is clipped high and low.
23	ALL_HOLES	Result invalid. Data contains all holes.
24	NO_DATA	Result invalid. No data on screen.
25	CURSOR_OFF_SCREEN	Result invalid. Cursor is not on screen.
26	MEASURE_CANCELLED	Result invalid. Measurement aborted.
27	MEASURE_TIMEOUT	Result invalid. Measurement timed-out.
28	NO_MEAS	Result invalid. No measurement to track.

RISetime

Command

```
:MEASure:RISetime [<source>]
```

The MEASure:RISetime command measures the rise time of the first displayed edge by measuring the time at the lower threshold of the rising edge, measuring the time at the upper threshold of the rising edge, then calculating the rise time with the following algorithm:

Rise time = time at upper threshold point − time at lower threshold point.

Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the RISETIME command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

With standard measurements selected, the lower threshold is at the 10% point and the upper threshold is at the 90% point on the rising edge.

Example

The following example measures the rise time of the displayed signal.

```
10  OUTPUT 707;":MEASURE:RISETIME"
20  END
```

Query

```
:MEASure:RISetime?[<source>]
```

The MEASure:RISetime query returns the rise time of the specified source.

Returned Format

```
[:MEASure:RISetime] <value>[,<result_state>]<NL>
```

`<value>` Rise time in seconds.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current value of rise time in the numeric variable, Rise, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:RISETIME?"
30  ENTER 707;Rise
40  PRINT Rise
50  END
```

SCRatch

Command

`:MEASure:SCRatch`

The MEASure:SCRatch command clears the measurement results from the screen.

Example

The following example clears the current measurement results from the screen.

```
10  OUTPUT 707;":MEASURE:SCRATCH"
20  END
```

SENDvalid

Command

```
:MEASure:SENDvalid {{ OFF |0} | {ON | 1}}
```

The SENDvalid command enables the result state code to be returned with the :MEASure:RESults? query.

Example

The following example turns send valid function on.

```
10  OUTPUT 707;":MEASURE:SENDVALID ON"
20  END
```

Query

```
:MEASure:SENDvalid?
```

The query returns the state of the Sendvalid control.

Returrned Format

```
{:MEASure:SENDvalid] {0 | 1}<NL>
```

Example

The following example places the current mode for sendvalid in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]                    !Dimension variable
20  OUTPUT 707;":MEASURE:SENDVALID?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

Refer to the MEASure:RESults query for information on the results returned and how they are affected by the SENDvalid command. Refer to the individual measurements for information on how the result state is returned.

SOURce

Command

```
:MEASure:SOURce <source>[,<source>]
```

The MEASure:SOURce command selects the source for measurements. Two sources can be specified with this command. All measurements except MEASURE:DELTATIME are made on the first specified source. The delta time measurement uses two sources if two are specified. If only one source is specified, the delta time measurement uses that source for both of its parameters.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number> | HISTogram | FFT}

`<number>` For channels: the number represents an integer, 1 through 4, followed by an optional letter, A or B (chan1 = chan1A). The integer is the slot in which the channel resides. The letter is used to identify which of two possible channels in the slot is being referenced.
For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example selects channel 1 as the source for measurements.

```
10  OUTPUT 707;":MEASURE:SOURCE CHANNEL1"
20  END
```

Query

```
:MEASure:SOURce?
```

The MEASure:SOURce query returns the current source selection.

Returned Format

```
[:MEASure:SOURce] <source>[,<source>]<NL>
```

Example

The following example places the currently specified sources in the string variable, Source$, then prints the contents of the variable to the controller's screen.

```
10  DIM Source$[50]           !Dimension variable
20  OUTPUT 707;":MEASURE:SOURCE?"
30  ENTER 707;Source$
40  PRINT Source$
50  END
```

STATistics

Command

```
:MEASure:STATistics {{OFF | 0} | {ON | 1}}
```

The MEASure:STATistics command turns the statistics measurements on and off. The statistics state only affects the MEASure:RESults query.

There are three modes for statistics on the front panel: off, mean/standard deviation, and min/max. When statistics is turned on over the bus, the last mode (mean/standard deviation or min/max) selected from the front panel is displayed on screen along with the current measurements.

Example

The following example turns the statistics function on.

```
10  OUTPUT 707;":MEASURE:STATISTICS ON"
20  END
```

Query

```
:MEASure:STATistics?
```

The :MEASure:STATistics query returns the current statistics mode.

Returned Format

```
[:MEASure:STATistics] {0 | 1}<NL>
```

Example

The following example places the current mode for statistics in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]                    !Dimension variable
20  OUTPUT 707;":MEASURE:STATISTICS?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

Refer to the MEASure:RESults query for information on the result returned and how they are affected by the STATistics command.

TEDGe

Command

```
MEASure:TEDGe <meas_thres_txt>,<slope><occurrence>
[,<source>]
```

This command measures the time interval between the trigger event and the specified threshold level and transition.

Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the TEDGecommand.

Query

```
:MEASure:TEDGe? <meas_thres_txt>,
<slope><occurrence> [,<source>]
```

The MEASure:TEDGe query returns the time interval between the trigger event and the specified threshold level and transition.

`<meas_thres_txt>` UPPer, MIDDle, or LOWer to identify the threshold.

`<slope>` {- (minus) for falling | + (plus) or optional for rising}.

`<occurrence>` numeric value representing which edge occurrence.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:TEDGe] <time>[,<result_state>]<NL>
```

`<time>` The time interval between the trigger event and the specified voltage level and transition.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the time interval between the trigger event and the 90% threshold on the second rising edge of the source waveform to the numeric variable, Time. The contents of the variable are then printed to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TEDGE? UPPER,+2"
30  ENTER 707;Time
40  PRINT Time
50  END
```

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TMAX

Command

```
MEASure:TMAX [<source>]
```

This command measures the first time at which the maximum voltage of the source waveform occurred. When FFT is the specified source, the frequency at which the first maximum value occurred is measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the TMAX command.

Query

```
:MEASure:TMAX? [<source>]
```

The MEASure:TMAX query returns the time at which the first maximum voltage occurred.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:TMAX] <time>[,<result_state>]<NL>
```

<time> Time at which the first maximum voltage occurred.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the time at which the first maximum voltage occurred to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TMAX?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TMIN

Command

`MEASure:TMIN [<source>]`

This command measures the time at which the first minimum voltage occurred. When FFT is the specified source, the frequency at which the first minimum value occurred is measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the TMIN command.

Query

`:MEASure:TMIN? [<source>]`

The MEASure:TMIN query returns the time at which the first minimum voltage occurred.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Returned Format

`[:MEASure:TMIN] <time>[,<result_state>]<NL>`

`<time>` Time at which the first minimum voltage occurred.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the time at which the first minimum voltage occurred to the numeric variable, Time, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TMIN?"
30  ENTER 707;Time
40  PRINT Time
50  END
```

> When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

TVOLt

Command

```
MEASure:TVOLt <voltage>,<slope><occurrence>
[,<source>]
```

This command measures the time interval between the trigger event and the defined voltage level and transition. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the TVOLt command. .

Query

```
:MEASure:TVOLt? <voltage>,<slope><occurrence>
[,<source>]
```

The MEASure:TVOLt query returns the time interval between the trigger event and the specified voltage level and transition.

<voltage> Voltage level at which time will be measured.

<slope> The direction of the waveform change when the the specified voltage is crossed - rising (+) or falling (−).

<occurrence> The number of the crossing to be reported (if one, the first crossing is reported; if two, the second crossing is reported, and so on).

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Returned Format

```
[:MEASure:TVOLt] <time>[,<result_state>]<NL>
```

<time> The time interval between the trigger event and the specified voltage level and transition.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the time interval between the trigger event and the transition through − .250 Volts on the third rising edge of the source waveform to the numeric variable, Time. The contents of the variable are then printed to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:TVOLT? -.250,+3"
30  ENTER 707;Time
40  PRINT Time
50  END
```

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

VAMPlitude

Command

```
:MEASure:VAMPlitude [<source>]
```

The MEASure:VAMPlitude command calculates the difference between the top and base voltage of the specified source. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VAMPlitude command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example calculates the difference between the top and base voltage of the specified source.

```
10  OUTPUT 707;":MEASURE:VAMPLITUDE"
20  END
```

Query

```
:MEASure:VAMPlitude? [<source>]
```

The MEASure:VAMPlitude query returns the calculated difference between the top and base voltage of the specified source.

Returned Format

```
[:MEASure:VAMPlitude] <value>[,<result_state>]<NL>
```

`<value>` Calculated difference between the top and base voltage.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current Vamplitude value in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VAMPLITUDE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VAVerage

Command

```
:MEASure:VAVerage  {CYCLe | DISPlay},[<source>]
```

The MEASure:VAVerage command calculates the average voltage over the displayed waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VAVerage command.

`CYCLe` The CYCLe parameter instructs the average measurement to measure the average voltage across the first period on the display.

`DISPlay` The DISPlay parameter instructs the average measurement to measure all the data on the display.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example calculates the average voltage over the displayed waveform.

```
10  OUTPUT 707;":MEASURE:VAVERAGE DISPLAY"
20  END
```

Query

```
:MEASure:VAVerage?{CYCLe | DISPlay},[<source>]
```

The MEASure:VAVerage query returns the calculated average voltage of the specified source. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VAVerage command.

Returned Format

```
[:MEASure:VAVerage] <value>[,<result_state>]<NL>
```

`<value>` The calculated average voltage.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current average voltage in the numeric variable, Average, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VAVERAGE? DISPLAY"
30  ENTER 707;Average
40  PRINT Average
50  END
```

VBASe

Command

```
:MEASure:VBASe [<source>]
```

The MEASure:VBASe command measures the statistical base of the waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VBASe command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the voltage at the base of the waveform.

```
10  OUTPUT 707;":MEASURE:VBASE"
20  END
```

Query

```
:MEASure:VBASe? [<source>]
```

The MEASure:VBASe query returns the measured voltage value at the base of the specified source.

Returned Format

```
[:MEASure:VBASe] <value>[,<result_state>]<NL>
```

`<value>` Voltage at the base of the waveform.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the current voltage at the base of the waveform to the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VBASE?"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VLOWer

Command

```
:MEASure:VLOWer [<source>]
```

The VLOWer command measures the voltage value at the lower threshold of the waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VLOWer command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Query

```
:MEASure:VLOWer?
```

The query returns the measured lower threshold of the selected source.

Returned Format

```
[:MEASure:VLOWer] <value>[,<result_state>]<NL>
```

`<value>` Voltage value at the lower threshold

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

xample

The following example returns the measured voltage at the lower threshold of the waveform to the numeric variable, Vlower, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VLOW?"
30  ENTER 707;Vlower
40  PRINT Vlower
50  END
```

VMAX

Command

```
:MEASure:VMAX [<source>]
```

The MEASure:VMAX command measures the absolute maximum voltage present on the selected source waveform. When FFT is the specified source, the maximum value in the spectrum is measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VMAX command.

`<source>` (CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT)

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the absolute maximum voltage on the waveform.

```
10  OUTPUT 707;":MEASURE:VMAX"
20  END
```

Query

```
:MEASure:VMAX? [<source>]
```

The MEASure:VMAX query returns the measured absolute maximum voltage present on the selected source waveform.

Returned Format

```
[:MEASure:VMAX] <value>[,<result_state>]<NL>
```

`<value>` Absolute maximum voltage present on the waveform.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured absolute maximum voltage on the waveform to the numeric variable, Maximum, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMAX?"
30  ENTER 707;Maximum
40  PRINT Maximum
50  END
```

VMIDdle

Command

```
:MEASure:VMIDdle [<source>]
```

The MEASure:VMID command measures the voltage level at the middle threshold of the waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VMID command.

Query

```
:MEASure:VMIDdle? [<source>]
```

This query returns the voltage value at the middle threshold of the waveform.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Returned Format

```
[MEASure:VMIDdle] <value>[,<result_state>]<NL>
```

<value> The middle voltage present on the waveform.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured middle voltage on the waveform to the numeric variable, Middle, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMID?"
30  ENTER 707;Middle
40  PRINT Middle
50  END
```

VMIN

Command

```
:MEASure:VMIN [<source>]
```

The MEASure:VMIN command measures the absolute minimum voltage present on the selected source waveform. When FFT is the specified source, the minimum value in the spectrum is measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VMIN command.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the absolute minimum voltage on the waveform.

```
10  OUTPUT 707;":MEASURE:VMIN"
20  END
```

Query

```
:MEASure:VMIN? [<source>]
```

The MEASure:VMIN query returns the measured absolute minimum voltage present on the selected source waveform.

Returned Format

```
[:MEASure:VMIN] <value>[,<result_state>]<NL>
```

<value> Absolute minimum voltage present on the waveform.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example returns the measured absolute minimum voltage on the waveform to the numeric variable, Minimum, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VMIN?"
30  ENTER 707;Minimum
40  PRINT Minimum
50  END
```

VPP

Command

```
:MEASure:VPP [<source>]
```

The MEASure:VPP command measures the maximum and minimum voltages on the selected source, then calculates the peak-to-peak voltage as the difference between the two voltages.When FFT is the specified source, the range of values in the spectrum are measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VPP command.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the peak-to-peak voltage of the previously selected source.

```
10  OUTPUT 707;":MEASURE:VPP"
20  END
```

Query

```
:MEASure:VPP? [<source>]
```

The MEASure:VPP query returns the peak-to-peak voltage of the specified source.

Returned Format

```
[:MEASure:VPP] <value>[,<result_state>]<NL>
```

<value> Peak-to-peak voltage of the selected source.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current peak-to-peak voltage in the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VPP?"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VRMS

Command

```
:MEASure:VRMS {CYCLe | DISPlay}, {AC | DC}
[,<source>]
```

The MEASure:VRMS command measures the RMS voltage of the selected waveform by subtracting the average value of the waveform from each data point on the display. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VRMS command.

`CYCLe` The CYCLE parameter instructs the RMS measurement to measure the RMS voltage across the first period of the display.

`DISPlay` The DISPLAY parameter instructs the RMS measurement to measure all the data on the display. Generally, RMS voltage is measured across one period, however, measuring multiple periods may be accomplished with the DISPLAY option. The DISPlay parameter is also useful when measuring noise.

`AC` The AC parameter is used to measure the RMS voltage subtracting out the DC component.

`DC` The DC parameter is used to measure RMS voltage including the DC component.

The AC RMS, DC RMS, and VAVG parameters are related as in the following formula:

$$DCVRMS^2 = ACVRMS^2 + VAVG^2$$

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the RMS voltage of the previously selected waveform.

```
10  OUTPUT 707;":MEASURE:VRMS CYCLE,AC"
20  END
```

Query

```
:MEASure:VRMS? {CYCLe | DISplay},{AC | DC}
[,<source>]
```

The MEASure:VRMS query returns the RMS voltage of the specified source.

Returned Format

```
[:MEASure:VRMS] <value>[,<result_state>]<NL>
```

`<value>` RMS voltage of the selected waveform.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the current AC RMS voltage over one period of the waveform in the numeric variable, Voltage, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VRMS? CYCLE,AC"
30  ENTER 707;Voltage
40  PRINT Voltage
50  END
```

VTIMe

Command

```
:MEASure:VTIMe? <time> [,<source>]
```

The MEASure:VTIMe command measures the voltage at the specified time. The time is referenced to the trigger event and must be on screen. When FFT is the specified source, the value at the specified frequency is measured. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VTIMe command.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number> | FFT}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.
For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

<time> Displayed time from trigger in seconds, or frequency in Hertz for FFT.

Query

```
:MEASure:VTIMe? <time> [,<source>]
```

The query returns the measured voltage.

Returned Format

```
[:MEASure:VTIMe] <value>[,<result_state>]<NL>
```

<value> Voltage at the specified time. When the source is FFT, the returned value is the vertical value at the horizontal setting passed in the VTIMe <time> parameter. The time parameter when FFT is the source is in Hertz.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the voltage at 500 ms in the numeric variable, Value, then prints the contents to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VTIME? 500E-3"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VTOP

Command `:MEASure:VTOP [<source>]`

The MEASure:VTOP command measures the statisical top of the selected source waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VTOP command.

`<source>` {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

`<number>` For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.
For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example The following example measures the voltage at the top of the waveform.

```
10  OUTPUT 707;":MEASURE:VTOP"
20  END
```

Query `:MEASure:VTOP? [<source>]`

The MEASure:VTOP query returns the measured voltage at the top of the specified source.

Returned Format `[:MEASure:VTOP] <value>[,<result_state>]<NL>`

`<value>` Voltage at the top of the waveform.

`<result_state>` If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the value of the voltage at the top of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VTOP?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

VUPper

Command

```
:MEASure:VUPper [<source>]
```

This command measures the voltage value at the upper threshold of the waveform. Sources are specified with the MEASURE:SOURCE command or with the optional parameter following the VUPper command.

<source> {CHANnel<number> | FUNCtion<number> | WMEMory<number>}

<number> For channels, this value is dependent on the type of plug-in and its location in the oscilloscope.

For functions: 1 or 2.
For waveform memories (WMEMORY): 1, 2, 3, or 4.

Example

The following example measures the voltage at the upper threshold of the waveform.

```
10  OUTPUT 707;":MEASURE:VUPper"
20  END
```

Query

```
:MEASure:VUPper? [<source>]
```

The query returns the measured upper threshold value of the selected source.

Returned Format

```
[:MEASure:VUPper] <value>[,<result_state>]<NL>
```

<value> Voltage at the upper threshold.

<result_state> If SENDVALID is ON, the result state is returned with the measurement result. See table 19-3, with the MEASure:RESults command, for a list of the result states.

Example

The following example places the value of the voltage at the upper threshold of the waveform in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":MEASURE:VUPPER?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

20

Pixel Memory Commands

Pixel Memory Commands

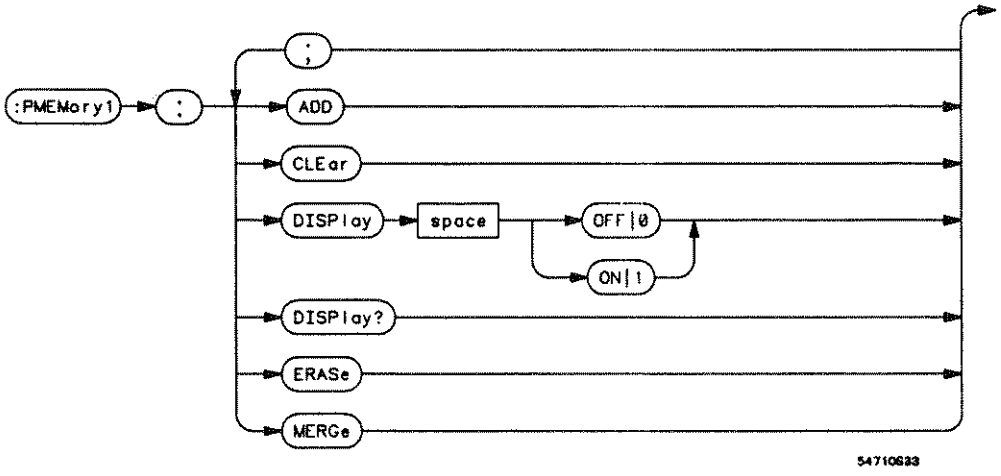
The Pixel Memory subsystem commands control the functions associated with the pixel memory. This allows merging (add to memory), clearing and turning the pixel memory display on and off. The Pixel Subsystem consists of the following commands and queries:

- ADD,
- CLEar,
- DISPlay,
- ERASe, and
- MERGe.

Refer to DISPlay:DATa for information on reading and writing pixel memory. Figure 20-1 is the syntax drawing of the Pixel Memory Subsystem.

Pixel Memory Syntax Diagram

Figure 20-1



ADD

Command

```
:PMEMory1:ADD
```

The PMEMory1:ADD command merges the current waveform(s) to the pixel memory. This is identical to the :PMEMory1:MERGe command.

CLEar

Command

```
:PMEMory1:CLEar
```

The PMEMory1:CLEar command clears the display memory. This is identical to the :PMEMory1:ERASe command.

DISPlay

Command

```
:PMEMory1:DISPlay {{ON| 1} | {OFF | 0}}
```

The PMEMory1:DISPlay command determines whether or not the pixel memory is viewable.

Query

```
:PMEMory1:DISPlay?
```

The query returns the state of the pixel memory.

Returned format:

```
[PMEMory1:DISPlay]     {1 | 0}
```

ERASe

Command

```
:PMEMory1:ERASe
```

The PMEMory1:ERASe command clears the pixel memory.

MERGe

Command

```
:PMEMory1:MERGe
```

The PMEMory1:MERGe command merges the current waveform(s) to the pixel memory.

21

Timebase Commands

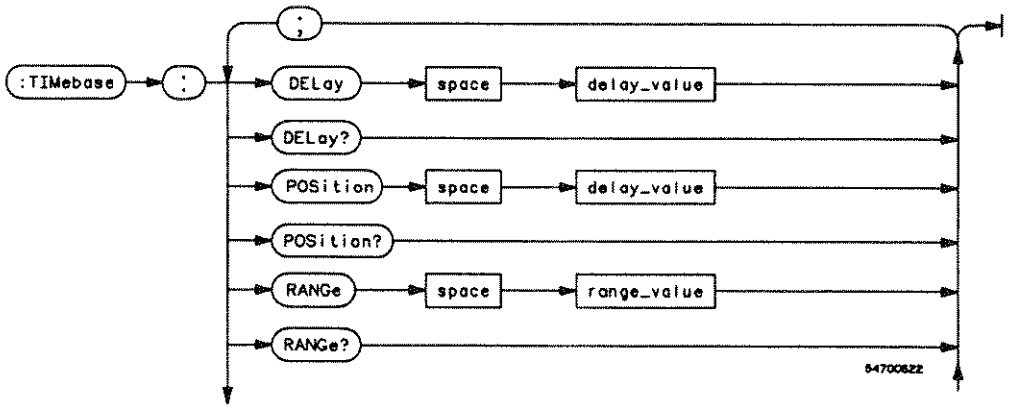
Timebase Commands

The TIMEBASE subsystem commands control the horizontal (x axis) oscilloscope functions. Figure 21-1 is the syntax diagram for the Timebase subsystem.

The Timebase subsystem contains the following commands and queries:

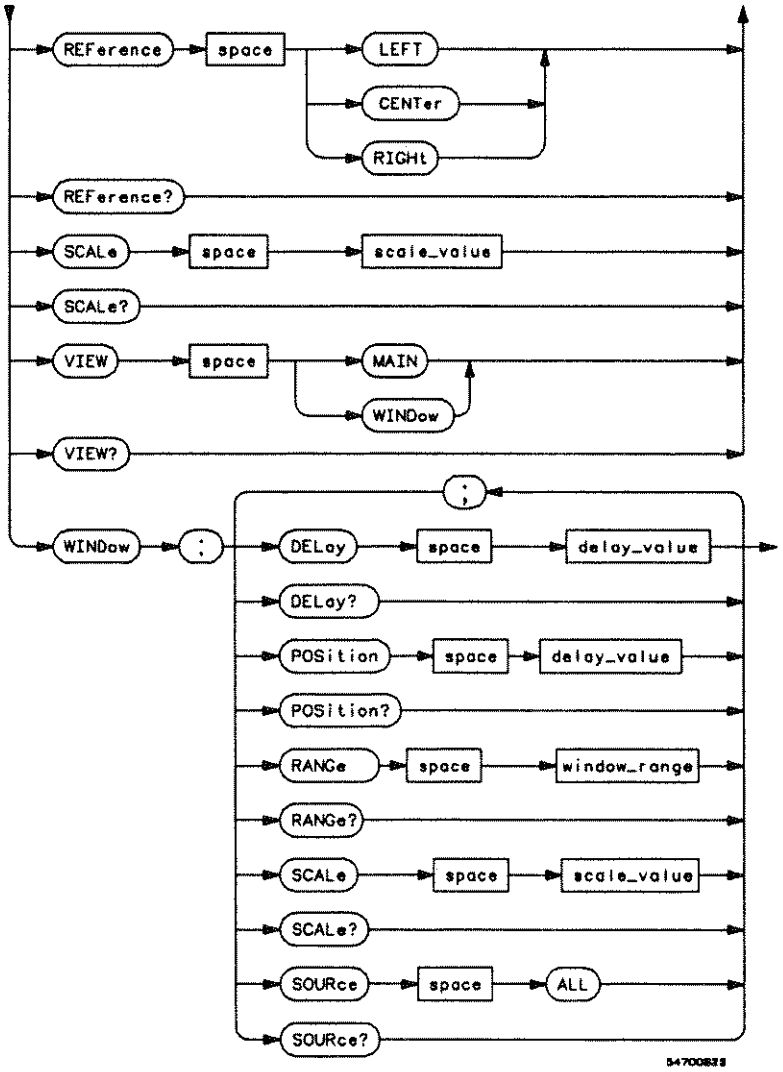
- DELay,
- POSition,
- RANGe,
- REFerence,
- SCALe,
- VIEW, and
- WINDow.

Figure 21-1



Timebase Subsystem Syntax Diagram

Figure 21-1 (continued)



Timebase Subsystem Syntax Diagram (continued)

DELay

Command

```
:TIMebase:DELay <delay_value>
```

The TIMebase:DELay command sets the time interval between the trigger event and the on-screen delay reference point. The delay reference point is set with the TIMEBASE:REFERENCE command.

> This command is the same as the TIMebase:POSition command and is provided for compatibility with previous instruments. The preferred command for compatibility with future instruments is TIMebase:POSition.

<delay_value>

Time in seconds from trigger to the on-screen delay reference point. The maximum value depends on the time/division setting.

Example

The following example sets the delay to 2 ms.

```
10  OUTPUT 707;":TIMEBASE:DELAY 2MS"
20  END
```

Query

```
:TIMebase:DELay?
```

The TIMebase:DELay query returns the current delay value.

Returned Format

```
[:TIMebase:DELay] <delay_value><NL>
```

Example

The following example places the current delay value in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:DELAY?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

When receiving numeric data into numeric variables, turn off the headers. Otherwise, the headers may cause misinterpretation of returned data.

See Also

The TIMEbase:POSition command performs the same function as this command and is the preferred command for new programs.

POSition

Command

```
:TIMebase:POSition <delay_value>
```

The TIMebase:POSition command sets the time interval between the trigger event and the on-screen delay reference point. The delay reference point is set with the TIMEbase:REFerence command.

> This command is the same as the TIMebase:DELay command and is preferred for compatibility with future instruments.

<delay_value>

Time in seconds from trigger to the on-screen delay reference point. The maximum value depends on the time/division setting.

Example

The following example sets the delay position to 2 ms.

```
10  OUTPUT 707;":TIMEBASE:POSITION 2MS"
20  END
```

Query

```
:TIMebase:POSition?
```

The TIMebase:POSition query returns the current delay value.

Returned Format

```
[:TIMebase:POSition] <delay_value><NL>
```

Example

The following example places the current delay value in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:POSITION?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

RANGe

Command

```
:TIMebase:RANGe <full-scale_range>
```

The TIMebase:RANGe command sets the full-scale horizontal time in seconds. The range value is ten times the time per division value and covers the range of 1 ns (100 ps/div) to 10 seconds (1 s/div).

<full_scale_range> 1 ns to 100 seconds.

Example

The following example sets the full-scale horizontal range to 10 ms.

```
10  OUTPUT 707;":TIMEBASE:RANGE 10E-3"
20  END
```

Query

```
:TIMebase:RANGe?
```

The TIMebase:RANGe query returns the current full-scale horizontal time.

Returned Format

```
[:TIMebase:RANGe] <full_scale_range><NL>
```

Example

The following example places the current full-scale horizontal range value in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

REFerence

Command

```
:TIMebase:REFerence {LEFT | CENTer | RIGHt}
```

The TIMebase:REFerence command sets the delay reference to the left, center, or right side for the main trace.

Example

The following example sets the delay reference to the center of the main trace.

```
10  OUTPUT 707;":TIMEBASE:REFERENCE CENTER"
20  END
```

Query

```
:TIMebase:REFerence?
```

The TIMebase:REFerence query returns the current delay reference for the main trace.

Returned Format

```
[:TIMebase:REFerence] {LEFT | CENTer | RIGHt}<NL>
```

Example

The following example places the current setting for the delay reference in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":TIMEBASE:REFERENCE?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

SCALe

Command

```
:TIMEbase:SCALe <time>
```

The TIMEbase:SCALe command sets the time base scale.

<time> Is the time value (in seconds per division).

Example

The following example sets the scale to 10 ms.

```
10  OUTPUT 707;":TIMEBASE:SCALE 10E-3"
20  END
```

Query

```
:TIMebase:SCALe?
```

The TIMebase:SCALe query returns the current scale time setting.

Returned Format

```
[:TIMebase:SCALe] <time><NL>
```

Example

The following example places the current scale value in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"        !Response headers off
20  OUTPUT 707;":TIMEBASE:RANGE?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

VIEW

Command

```
:TIMebase:VIEW {MAIN | WINDow | INTensified}
```

The TIMebase:VIEW command turns the windowed view on and off. If the windowed view is on, the main waveforms are not shown.

INTensified displays the main waveform with the windowed view intensified.

Example

The following example turns the windowed view on.

```
10  OUTPUT 707;":TIMEBASE:VIEW WINDOW"
20  END
```

Query

```
:TIMebase:VIEW?
```

The TIMebase:VIEW query returns the current state of the TIMebase:VIEW command.

Returned Format

```
[:TIMebase:VIEW] {MAIN | WINDow | INTensified}<NL>
```

Example

The following example places the current state of windows in the string variable, State$, then prints the contents of the variable to the controller's screen.

```
10  DIM State$[50]                    !Dimension variable
20  OUTPUT 707;":TIMEBASE:VIEW?"
30  ENTER 707;State$
40  PRINT State$
50  END
```

WINDow:DELay

Command

```
:TIMebase:WINDow:DELay <delay_position>
```

The TIMebase:WINDow:DELay sets the position of the time-base window on the main sweep. The range for this command is determined by the main sweep range (or size) and the main sweep delay value. The value for this command must keep the time-base window within the main sweep range.

<delay_position> Time in seconds from trigger to on-screen delay reference point. Maximum depends on the main sweep range (or size) and the main sweep delay value.

Example

The following example sets the time-base window delay position to 20 ns.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:DELAY 20NS"
20  END
```

Query

```
:TIMebase:WINDow:DELay?
```

The TIMebase:WINDow:DELay query returns the current position of the time base window.

Returned Format

```
[:TIMebase:WINDow:DELay] <delay_position><NL>
```

Example

The following example places the current position of the time base window in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:DELAY?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

See Also

The TIMEbase:WINDow:POSition command performs the same function as this command and is the preferred command for new programs.

WINDow:POSition

Command

```
:TIMebase:WINDow:POSition <delay position>
```

The TIMebase:WINDow:POSition sets the position of the time-base window on the main sweep. The range for this command is determined by the main sweep range (or size) and the main delay value. The value for this command must keep the time-base window on the main sweep range.

<delay_position> Time in seconds from trigger to onscreen delay reference point. Maximum depends on the main sweep range (or size) and the main sweep delay value.

Example

The following example sets the time-base window delay position to 20 ns.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:POSITION 20NS"
20  END
```

Query

```
:TIMebase:WINDow:POSition?
```

The TIMebase:WINDow:POSition query returns the current position of the time-base window.

Returned Format

```
[:TIMebase:WINDow:POSition] <delay_position><NL>
```

Example

The following example places the current position of the time-base window in the numeric variable, Setting, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:POSITION?"
30  ENTER 707;Setting
40  PRINT Setting
50  END
```

WINDow:RANGe

Command

```
:TIMebase:WINDow:RANGe <full_scale_range>
```

The TIMebase:WINDow:RANGe command sets the full-scale range of the time-base window. The range value is ten times the time per division of the window. The maximum range of the window is the current main range. The minimum window range is 10 ps (1 ps/div).

<full_scale_range> The full-scale range of the time base window.

Example

The following example sets the full-scale range of the time base window to 100 ns.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:RANGE 100NS"
20  END
```

Query

```
:TIMebase:WINDow:RANGe?
```

The TIMebase:WINDow:RANGe query returns the current full-scale range of the time base window.

Returned Format

```
[:TIMebase:WINDow:RANGe] <full_scale_range><NL>
```

Example

The following example reads the current full-scale range of the time base window into the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":TIMEBASE:WINDOW:RANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

WINDow:SOURce

Command

```
:TIMebase:WINDow:SOURce ALL
```

The TIMebase:WINDow:SOURce command selects the source for the time-base window. The only parameter available for this command is ALL, which selects all sources displayed on the main screen.

Example

The following example selects all of the sources displayed on the main screen for the sources for the current window.

```
10  OUTPUT 707;":TIMEBASE:WINDOW:SOURCE ALL"
20  END
```

Query

```
:TIMebase:WINDow:SOURce?
```

The TIMebase:WINDow:SOURce query returns the current source for the window.

Returned Format

```
[:TIMebase:WINDow:SOURce] ALL<NL>
```

Example

The following example places the current setting for the window source in the string variable, Current$, then prints the contents of the variable to the controller's screen.

```
10  DIM Current$[50]          !Dimension variable
20  OUTPUT 707;":TIMEBASE:WINDOW:SOURCE?"
30  ENTER 707;Current$
40  PRINT Current$
50  END
```

22

Trigger Commands

Trigger Commands

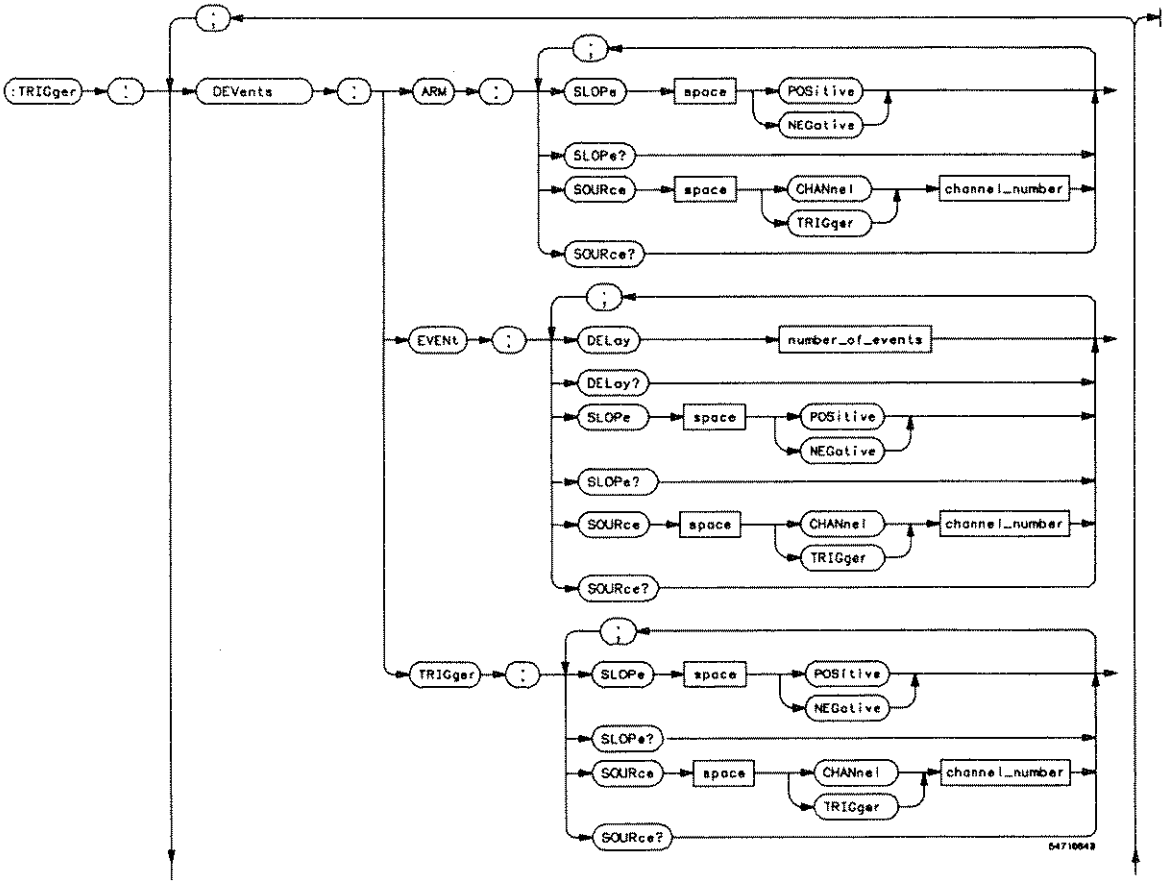
The commands in the TRIGger subsystem define the conditions for triggering. Many of the commands in the Trigger subsystem are used in more than one of the trigger modes. The command set has been defined to represent more closely what is in the front-panel trigger menus at the expense of some compatibility with the command sets for previous oscilloscopes. For this reason, the oscilloscope accepts some commands for compatibility with previous instruments. The alternative command that is accepted by the oscilloscope is noted for each command. The Trigger Subsystem consists of the following commands and queries:

- DEVents,
- DTIMe,
- EDGe,
- GLITch,
- HOLDoff,
- HYSTeresis,
- LEVel,
- MODE,
- PATTern,
- SLOPe,
- SOURce,
- STATe,
- STV,
- SWEep, and
- UDTV.

Figure 22-1 is the Trigger Subsystem commands syntax diagram.

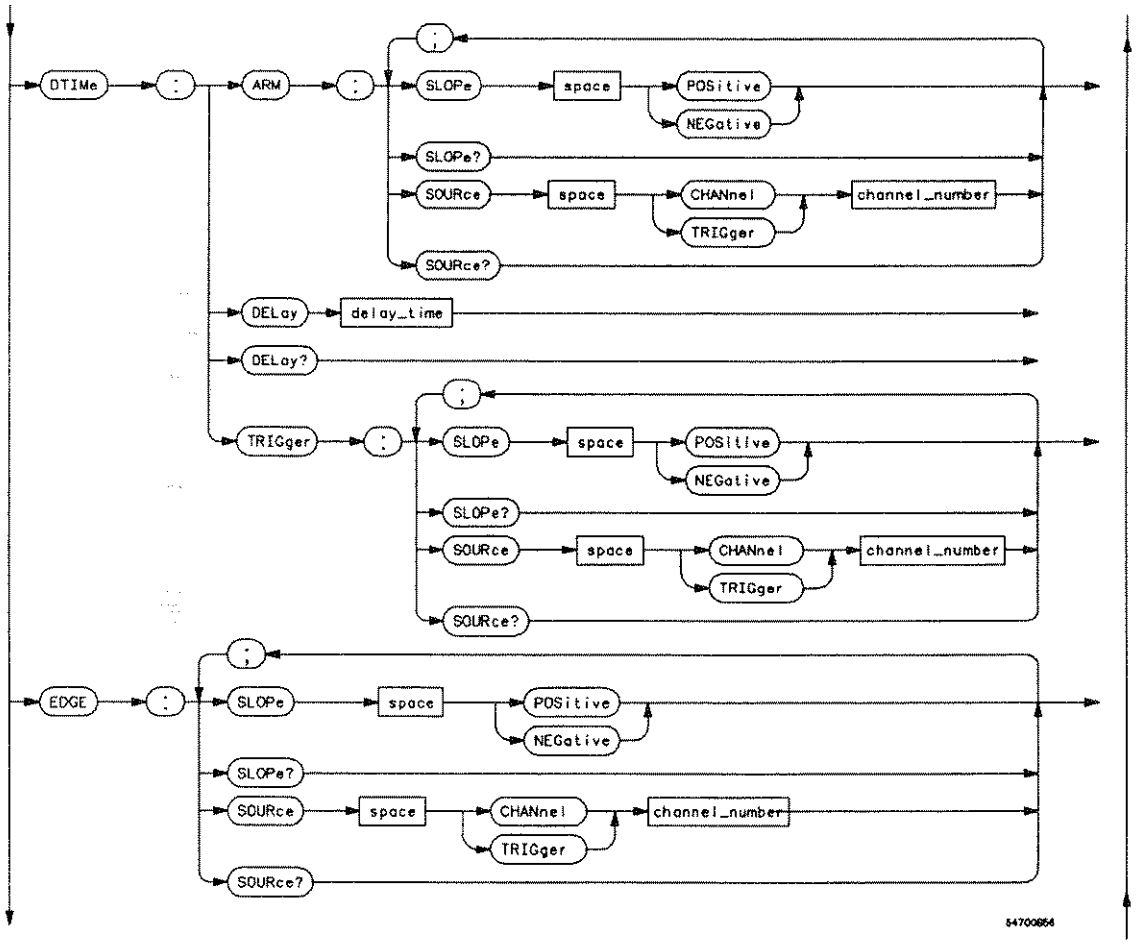
Table 22-1 is a list of the trigger commands. The commands are listed under the trigger modes that use them.

Figure 22-1



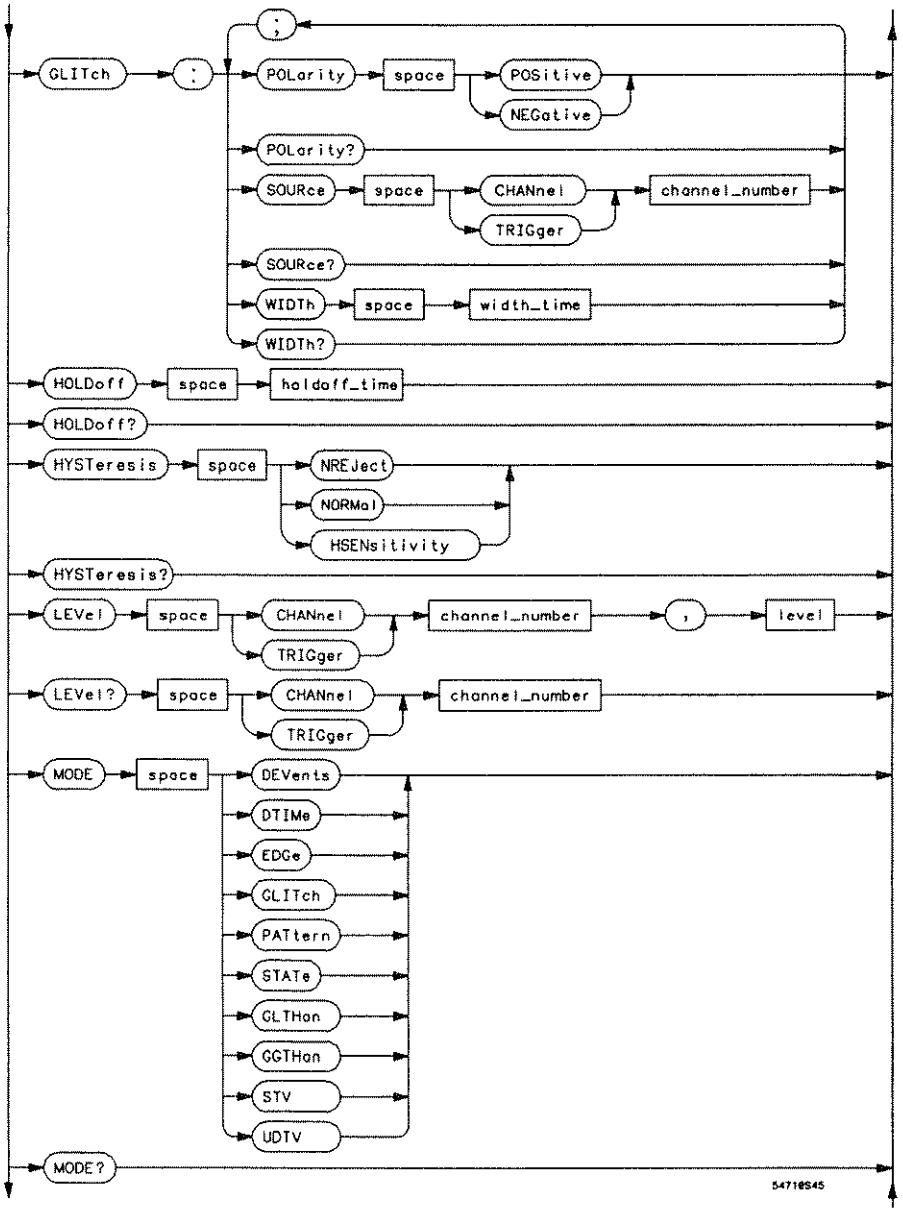
Trigger Subsystem Syntax Diagram

Figure 22-1 (continued)



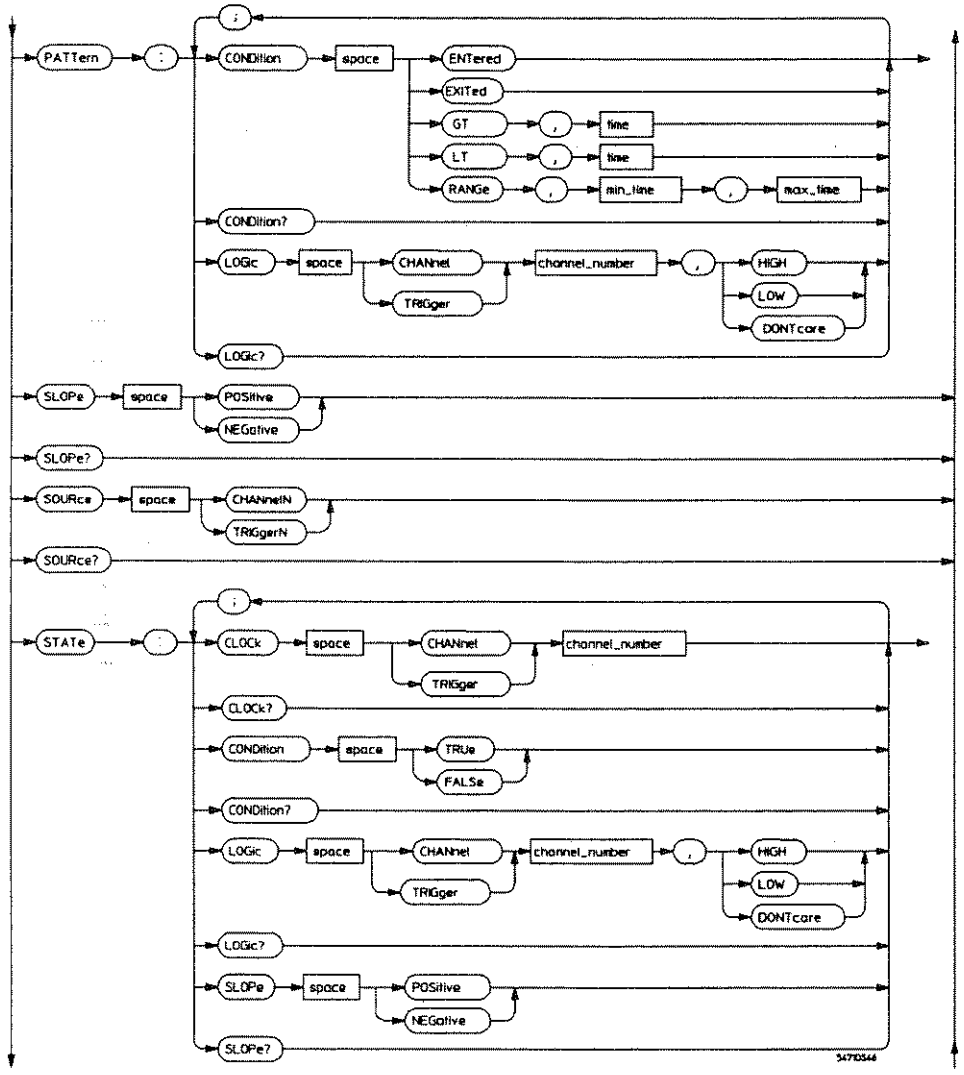
Trigger Subsystem Syntax Diagram (continued)

Figure 22-1 (continued)



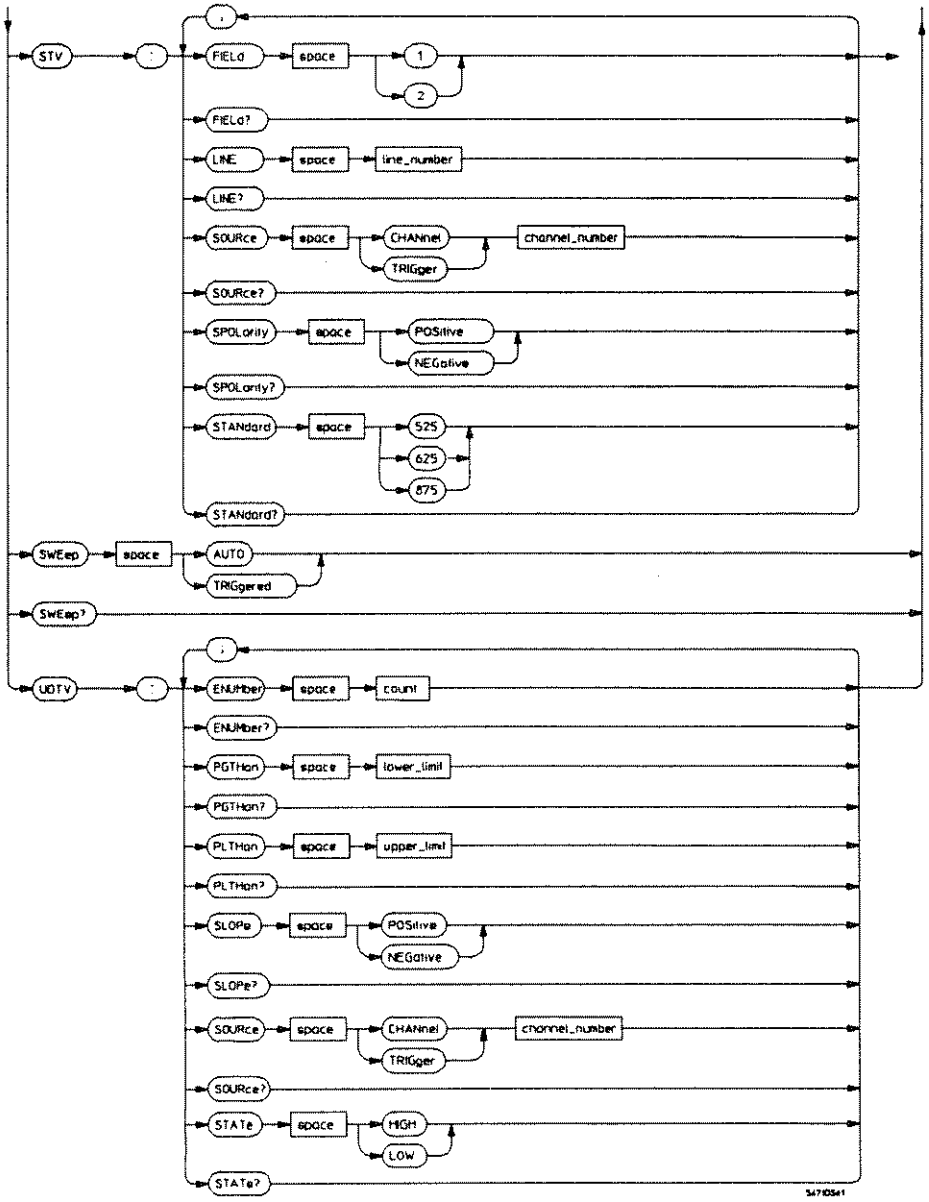
Trigger Subsystem Syntax Diagram (continued)

Figure 22-1 (continued)



Trigger Subsystem Syntax Diagram (continued)

Figure 22-1 (continued)



DEVents

Command

`:TRIGger:DEVents`

The commands in this subsystem select the parameters for the Event Delay Triggering mode. This mode must have been previously selected using the :TRIGger:MODE command.

> In this subsystem DEVents is the SCPI form of the header while :TRIGger:EDLY command and query are accepted by the oscilloscope to provide compatibility with previous instruments. The two forms can be used interchangeably in any of the TRIGer:DEVents commands and queries.

The :TRIGger:DEVents command is sent with one or more of the following commands and queries:

- ARM,
- EVENt, and
- TRIGger.

DEVents:ARM

Command

```
:TRIGger:DEVents:ARM
```

This command selects the parameters for the arming source for the Delay Event trigger mode.

> The TRIGger:EDLY:ARMcommand and query are accepted for compatibility with previous instruments.

The TRIGger:DEVents:ARM command is sent with one or more of the following commands and queries:

- SLOPe, and
- SOURce.

Command

```
:TRIGger:DEVents:ARM:SLOPe {POSitive | NEGative}
```

This command selects the slope for the source previously selected by the TRIGger:DEVents:ARM:SOURce command.

> The TRIGger:EDLY:ARM:SLOPe command and query are accepted for compatibility with previous intruments.

This also controls the slope of the Delay:TIME:Arm source.

Query

```
:TRIGger:DEVents:ARM:SLOPe?
```

The query returns the currently selected slope for the arming signal.

Returned Format

```
[:TRIGger:DEVents:ARM:SLOPe] {POS|NEG}<NL>
```

Command `:TRIGger:DEVents:ARM:SOURce {CHANnelN|TRIGgerN}`

The command selects the source for arming the trigger in the event delay mode.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

> TRIGgerN is the number of the slot where the external trigger is located.

> The TRIGger:EDLY:ARM:SOURce command and query are accepted for compatibility with previous instruments.

Query `:TRIGger:DEVents:ARM:SOURce?`

The query returns the currently selected source.

Returned Format `[:TRIGger:DEVents:ARM:SOURce] {CHANnelN|TRIGgerN}`

DEVents:EVENt

The :TRIGger:DEVents:EVENt command is sent with one or more of the following commands and queries:

- DELay,
- SLOPe, and
- SOURce.

Command `:TRIGger:DEVents:EVENt:DELay <time>`

The TRIGger:DEVents:EVENt:DELay command specifies the number of events to delay after arming the trigger before actually looking for the trigger event.

`<number>` Is the number of delay events. Allowable values range from 1 to 16,000,000.

> The TRIGger:EDLY:EVENt:DELay command and query are accepted for compatibility with previous instruments.

Query `:TRIGger:DEVents:EVENt:DELay?`

The query returns the currently programmed event delay value.

Returned Format `[:TRIGger:DEVents:EVENt:DELay] <time>`

Command `:TRIGger:DEVents:EVENt:SLOPe {POSitive | NEGative}`

The command specifies the slope of the previously selected event source to count with the TRIGger:DEVents:EVENt:DELay command.

> The TRIGger:EDLY:DELay command and query are accepted for compatibility with previous instruments.

Query `:TRIGger:DEVents:EVENt:SLOPe?`

The query returns the currently selected slope for event delay source.

Returned Format `[:TRIGger:DEVents:EVENt:SLOPe] {POSitive|NEGative}`

Command `:TRIGger:DEVevents:EVENt:SOURce {CHANnelN|TRIGgerN}`

This command selects the source for the events to be counted to satisify the delay criteria in the Event Delay trigger mode.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Query `:TRIGger:DEVents:EVENt:SOURce?`

The query returns the currently selected source for events.

Returned Format `[:TRIGger:DEVents:EVENt:SOURce] {CHANnelN|TRIGgerN}`

DEVents:TRIGger

Command

```
:TRIGger:DEVents:TRIGger
```

The :TRIGger:DEVents:TRIGger command sets the parameters for the actual trigger in the Event Delay trigger mode.

> The TRIGger:EDLY commands and queries are accepted for compatibilty with previous instruments.

The TRIGger:DEVents:TRIGger command is sent with one or more of the following commands and queries:

- SLOPe, and
- SOURce.

Command

```
:TRIGger:DEVents:TRIGger:SLOPe {POSitive |NEGative}
```

This command sets the slope of the channel selected by the previously issued :TRIGger:DEVents:TRIGger:SOURce command.

This command also controls the trigger slope for the Delay:TIMe trigger source.

Query

```
:TRIGger:DEVents:TRIGger:SLOPe?
```

The query returns the slope of the currently selected source.

Returned Format

```
[:TRIGger:DEVents:TRIGger:SLOPe] {POS | NEG}<NL>
```

Command

```
:TRIGger:DEVents:TRIGger:SOURce
{CHANnelN | TRIGgerN}
```

This command selects the source that generates the trigger in the Event Delay trigger mode.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Query

```
:TRIGger:DEVents:TRIGger:SOURce?
```

The query returns the currently selected trigger source for the Event Delay trigger mode.

Returned Format

```
[:TRIGger:DEVents:TRIGger:SOURce]    {CHANnelN|TRIGgerN}
```

DTIMe

Command

`TRIGger:DTIMe`

This command defines the conditions for the Delay Time Trigger mode. This mode must have been previously selected using the :TRIGger:MODE command.

> In this subsystem, DTIMe is the SCPI form of the header while TDLY is provided to provide compatibility with previous instruments. The two forms can be used interchangeably.

One or more of the following commands and queries are sent with the :TRIGger:DTIMe command:

- ARM,
- DELay, and
- TRIGger.

DTIMe:ARM

Command

TRIGger:DTIMe:ARM

This command sets the source and slope of the signal used to arm the trigger for the Time Delay trigger mode.

> The TRIGger:TDLY:ARM command and query are accepted for compatibility with previous instruments.

One or more of the following commands and queries are sent with the :TRIGger:DTIMe:ARM command:

- SLOPE, and
- SOURce.

Command

```
:TRIGger:DTIMe:ARM:SLOPe {POSitive|NEGative}
```

This command sets the slope of the trigger arming signal previously selected with the :TRIGger:DTIMe:ARM:SOURce command.

> The TRIGger:TDLY:ARM:SLOPe command and query are accepted for compatibilty with previous instruments.

This command also controls the slope for the Delay EVents mode Arm source.

Query

```
:TRIGger:DTIMe:ARM:SLOPe?
```

The query returns the slope of the currently selected source.

Returned Format

```
[:TRIGger:DTIMe:ARM:SLOPe] {POSitive | NEGative}
```

Command `:TRIGger:DTIMe:ARM:SOURce {CHANnelN | TRIGgerN}`

This command selects the source for the trigger arm signal in the Time Delay trigger mode.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

> The TRIGger:TDLY:ARM:SOURce command and query are accepted for compatibility with previous instruments.

Query `:TRIGger:DTIMe:ARM:SOURce?`

The query returns the currently selected source.

Returned Format `[:TRIGger:DTIMe:ARM:SOURce] {CHANnelN | TRIGgerN}`

DTIMe:DELay

Command

```
TRIGger:DTIMe:DELay <time>
```

This command specifies the time to delay after receiving the ARM signal before enabling the trigger in Time Delay trigger mode.

`<time>` Is the delay time from 30 ns to 160 ms.

> The TRIGger:TDLY:DELay command and query are accepted for compatibility with previous instruments.

Query

```
:TRIGger:DTIMe:DELay?
```

The query returns the currently programmed delay time.

Returned Format

```
[:TRIGger:DTIMe:DELay] <time>
```

DTIMe:TRIGger

Command `TRIGger:DTIMe:TRIGger`

This commands set the parameters for the trigger source in Time Delay trigger mode.

> The TRIGger:TDLY:TRIGger command and query are accepted for compatibility with previous instruments.

Command `TRIGger:DTIMe:TRIGger:SLOPe{POSitive | NEGative}`

This command specifies the slope that will generate the trigger on the signal previously selected by the :TRIGger:DTIMe:TRIGger:SOURce command.

This will also control the slope for the DEVents mode trigger source.

> The TRIGger:TDLY:TRIGger:SLOPe command and query is accepted for compatibility with previous instruments.

Query `:TRIGger:DTIMe:TRIGger:SLOPe?`

The query returns the currently selected slope.

Returned Format `[:TRIGger:DTIMe:TRIGger:SLOPe] {POS|NEG}`

Command

```
TRIGger:DTIMe:TRIGger:SOURce {CHANnelN|TRIGgerN}
```

This command selects the source for the trigger in the Time Delay trigger mode.

CHANnelN N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

TRIGgerN N, an integer 1 to 4, represents the slot where the external trigger resides.

> The TRIGger:TDLY:TRIGger:SOURce command and query are accepted for compatibility with previous instruments.

Query

```
:TRIGger:DTIMe:TRIGger:SOURce?
```

The query returns the currently selected trigger source.

Returned Format

```
[:TRIGger:DTIMe:TRIGger:SOURce] {CHANnelN|TRIGgerN}
```

EDGE

Command `:TRIGger:EDGE`

The TRIGger:EDGE commands control the source and slope conditions for the edge triggering mode.

The EDGE command is sent with one or more of the following commands and queries:

- SLOPe, and
- SOURce.

EDGE:SLOPe

Command

```
:TRIGger:EDGE:SLOPe {POSitive|NEGative}
```

The TRIGger:EDGE:SLOPe command sets the slope of the trigger source previously selected by the TRIGger:EDGE:SOURce or TRIGger:SOURce command.

Query

```
:TRIGger:EDGE:SLOPe?
```

The query returns the currently selected slope for the specified edge trigger source.

Returned Format

```
[:TRIGger:EDGE:SLOPe] {POS|NEG}
```

EDGE:SOURce

Command `:TRIGger:EDGE:SOURce {CHANnelN|TRIGgerN}`

The TRIGger:EDGE:SOURce command selects the source for edge mode triggering. This is the source that will be used for subsequent TRIGger:EDGE:SLOPe or TRIGger:SLOPe commands or queries.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` TRIGgerN, an integer 1 to 4, represents the slot where the external trigger resides.

Query `:TRIGger:EDGE:SOURce?`

The query returns the currently selected edge mode trigger source.

Returned Format `[:TRIGger:EDGE:SOURce]{CHANnelN|TRIGgerN}<NL>`

GLITch

Command

`:TRIGger:GLITch`

This command sets the conditions for the Glitch (GLITch), Glitch less than (GLTHan), and Glitch greater than (GGTHan) trigger modes. Note that both GLITch and GLTHan both refer to the Glitch less than mode. GLITch is still accepted for compatibility with programs written before the command set was enhanced to support the Glitch greater than mode. The :TRIGger:MODE command is used to select one of the glitch trigger modes.

The :TRIGger:GLITch command is sent with one or more of the following commands and queries:

- POLarity,
- SOURce, and
- WIDth.

GLITch:POLarity

Command

```
:TRIGger:GLITch:POLarity {POSitive | NEGative}
```

This command defines the polarity of the glitch as positive or negative. The trigger source must be set using the :TRIGger:GLITch:SOURce command *before* the polarity is set.

Query

```
:TRIGger:GLITch:POLarity?
```

The query returns the currently selected glitch polarity.

Returned Format

```
[:TRIGger:GLITch:POLarity]    {POS|NEG}
```

GLITch:SOURce

Command

```
:TRIGger:GLITch:SOURce {CHANnelN | TRIGgerN}
```

This command sets the source for the glitch trigger modes.

N represents an integer, 1 to 4, indicating the slot in which the source resides, followed by an optional letter, A or B, used when the source plug-in has more than one channel or external trigger input in the same slot. If no letter is specified, then A is assumed (i.e., CHAN1 = CHAN1A). Note that the external trigger inputs on some plug-ins may not be assigned as a source for the glitch trigger modes. For example, the HP 54711A has no logic triggers at all, and the HP 54722 can only use TRIGger4. The HP 54712, HP 54713, HP 54714, and HP 54721 all have logic triggering capability. Any triggering limitations are described in the plug-in manual.

Query

```
:TRIGger:GLITch:SOURce?
```

The query returns the currently selected source for the trigger glitch modes.

Returned Format

```
[:TRIGger:GLITch:SOURce]    {CHANnelN | TRIGgerN}
```

GLITch:WIDTh

Command `:TRIGger:GLITch:WIDTh <width>`

This command sets the maximum or minimum glitch width, depending on the trigger mode selected with the :TRIGger:MODE command (maximum for GLTHan, minimum for GGTHan).

`<width>` Glitch width from 3 ns to 160 ms.

Query `:TRIGger:GLITch:WIDTh?`

The query returns the currently specified glitch width.

Returned Format `[:TRIGger:GLITch:WIDTh] <width>`

HOLDoff

Command `:TRIGger:HOLDoff <holdoff>`

The TRIGger:HOLDoff command is valid in the EDGE, GLITCH, PATTERN or STATE trigger mode. It specifies the amount of time the scope should wait after receiving trigger before enabling the trigger again.

`<holdoff>` Is the holdoff time. Allowable values range from 60 ns to 320 ns.

Query `:TRIGger:HOLDoff?`

The query returns the current holdoff value for the current mode.

Returned Format `[:TRIGger:HOLDoff] <holdoff>`

HYSTeresis

Command

```
:TRIGger:HYSTeresis {NREJect|NORMal|HSENsitivity}
```

The TRIGger:HYSTeresis command specifies the trigger hysteresis. NREJ (Noise REJect) gives maximum hysteresis but lowest trigger bandwidth, NORM (NORMal) is the typical hysteresis selection, HSEN (High SENsitivity) gives minimum hysteresis and highest bandwidth.

Query

```
:TRIGger:HYSTeresis?
```

Returned Format

```
[TRIGger:HYSTeresis] {NREJect|NORMal|HSENsitivity}
```

LEVel

Command `:TRIGger:LEVel {CHANnelN|TRIGgerN},<level>`

The TRIGger:LEVel command is valid in all trigger modes: EDGE, GLITch, PATtern, STATe, DTIMe, and DEVents. It specifies the trigger level on the specified trigger source (for edge and glitch modes) or for a channel included in trigger specification (for pattern, state, time delay and event delay modes).

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Trigger level limits are plug-in dependent.

> There is only one trigger level stored in the instrument for each channel so it does not matter which trigger mode is selected.

Query `:TRIGger:LEVel? {CHANnelN|TRIGgerN}`

The query returns the trigger level for the selected source.

Returned Format `[:TRIGger:LEVel] {CHANnelN|TRIGgerN},<level>`

MODE

Command

```
:TRIGger:MODE {EDGE | GLITch | GLTHan | GGTHan |
PATTern | STATe | DEVents | DTIMe | STV | UDTV}
```

The TRIGger:MODE command selects the trigger mode. The available modes are:

Table 22-2 Trigger Mode Settings

Mode	Definition
EDGE	Edge trigger mode.
GLITch	Trigger on pulse whose width is less than a specified amount of time. Same as GLTHan.
GLTHan	Trigger on pulse whose width is less than a specified amount of time. Same as GLITch.
GGTHan	Trigger on pulse whose width is greater than a specified amount of time.
PATTern	Trigger on a specific bit pattern. May be qualified by time.
STATe	Pattern trigger that is qualified by a clock edge.
DEVents	Delay by events.
DTIMe	Delay by time.
STV (Standard TV)	TV trigger mode using standard pre-defined parameters.
UDTV (User Defined TV)	TV trigger mode using user-defined parameters.

GLITch and GLTHan may be used interchangeably. GLITch remains in the command set to provide compatibility with programs written before the GLTHan command was added.

For compatibility with programs written for other instruments, the 54710/20 will accept the string "EDLY" in place of "DEVents" and "TDLY" in place of "DTIMe." However, new programs should use the standard command set.

Query

```
:TRIGger:MODE?
```

The query returns the currently selected trigger mode.

Returned Format

```
[:TRIGger:MODE] {EDGE | GLTHan | GGTHan | PATTern | STATe |
DEVents | DTIMe | STV | UDTV}
```

GLTHan is returned when the MODE is specified to be either GLITch or GLTHan.

PATTern

Command

`:TRIGger:PATTern`

The :TRIGger:PATTern commands define the conditions for the Pattern trigger mode. This mode must be selected using the :TRIGger:MODE command before setting up the conditions.

The :TRIGger:PATTern command is sent with one or more of the following commands and queries:

- CONDition, and
- LOGic.

PATTern:CONDition

Command

```
:TRIGger:PATTern:CONDition {ENTer|EXIT|{GT,
<time>} | {LT, <time>} | {RANGe, <min_time>,
<max_time>}}
```

This command describes the condition applied to the trigger pattern to actually generate a trigger.

`<time>` Is the time in seconds.

Query

```
:TRIGger:PATTern:CONDition?
```

The query returns the currently defined trigger condition.

Returned Format

```
[:TRIGger:PATTern:CONDition] {ENTer | EXIT | {GT, <time>} |
{LT, <time>} | {RANGe, <time>}}<NL>
```

PATTern:LOGic

Command

```
:TRIGger:PATTern:LOGic {CHANnelN | TRIGgerN},{HIGH
| LOW | DONTcare}
```

This command defines the logic criteria for a selected channel for the Pattern trigger mode.

Query

```
:TRIGger:PATTern:LOGic? {CHANnelN | TRIGgerN}
```

The query returns the current logic criteria for a selected channel.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N is an integer 1 through 4 identifying the slot where the external trigger resides.

Returned Format

```
[:TRIGger:PATTern:LOGic] {CHANnelN | TRIGgerN}
{HIGH|LOW|DONTcare}
```

SLOPe

Command `:TRIGger:SLOPe {POSitive|NEGative}`

The TRIGger:SLOPe command specifies the slope of the edge on which to trigger in EDGE triggering mode. It sets the slope of the channel previously selected with the TRIGger:SOURce or TRIGger:EDGE:SOURce command. It is equivalent to the TRIGger:EDGE:SLOPe command and is included for compatibility with older instruments.

Query `:TRIGger:SLOPe?`

The query returns the current slope for the currently selected trigger mode.

Returned Format `[:TRIGger:SLOPe] {POSitive|NEGative}`

SOURce

Command `:TRIGger:SOURce {CHANnelN|TRIGgerN}`

The TRIGger:SOURce command selects the channel or trigger that produces the trigger in EDGE mode. This command also identifies the source for any subsequent SLOPE and PROBE commands for EDGE mode triggering. It is equivalent to the :TRIGger:EDGE:SOURce command and is included for compatibility with older instruments.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Query `:TRIGger:SOURce?`

The query returns the current trigger source of the current mode.

Returned Format `[:TRIGger:SOURce] {CHANnelN|TRIGgerN}`

STATe

Command

`:TRIGger:STATe`

The TRIGger:STATe command controls the conditions for the State trigger mode. The STATe mode must be selected by the :TRIGger:MODE command before setting these conditions.

The TRIGger:STATe command is sent with one or more of the following of commands and queries:

- CLOCk,
- CONDition,
- LOGic, and
- SLOPe.

STATe:CLOCk

Command `:TRIGger:STATe:CLOCk {CHANnelN | TRIGgerN}`

This command selects the source for the clock signal in the State trigger mode.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Query `:TRIGger:STATe:CLOCk?`

The query returns the current selected clock source.

Returned Format `[:TRIGger:STATe:CLOCk] {CHANnelN | TRIGgerN}`

STATe:CONDition

Command

```
:TRIGger:STATe:CONDition {TRUE|FALSe}
```

This command determines if a trigger is generated when the pattern speicified by the LOGic command is TRUE (present) or FALSe (not present).

Query

```
:TRIGger:STATe:CONDition?
```

The query returns the currently specified condition.

Returned Format

```
[:TRIGger:STATe:CONDition]    {TRUE|FALSe}
```

STATe:LOGic

Command

```
:TRIGger:STATe:LOGic {CHANnelN | TRIGgerN},
{LOW|HIGH|DONTcare}
```

This command defines the logic state of the specified channel for the State trigger mode.

Query

```
:TRIGger:STATe:LOGic? {CHANnelN | TRIGgerN}
```

The query returns the logic state definition for the specified channel or trigger input.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Returned Format

```
[:TRIGger:STATe:LOGic] {CHANnelN | TRIGgerN}
{LOW|HIGH|DONTcare}
```

`DONTcare` is returned on the channel currently selected as clock.

STATe:SLOPe

Command

```
:TRIGger:STATe:SLOPe {POSitive|NEGative}
```

This command specifies the slope of the input previously selected by the :TRIGger:STATe:CLOCk command.

Query

```
:TRIGger:STATe:SLOPe?
```

The query returns the currently defined slope for the clock in State trigger mode.

Returned Format

```
[:TRIGger:STATe:SLOPe] {POSitive | NEGative}
```

STV

Command

`:TRIGger:STV`

This command sets the conditions for the TV trigger mode using standard, pre-defined parameters.

This mode is used for triggering on clamped television signals, and allows you to select one of the TV signal frames and one of the lines within that frame.

The :TRIGger:MODE command is used to select the standard TV trigger mode.

The :TRIGger:STV command is sent with one or more of the following commands and queries:

- FIELd,
- LINE,
- SOURce,
- SPOLarity, and
- STANDard.

STV:FIELd

Command

```
:TRIGger:STV:FIELd {1 | 2}
```

The TRIGger:STV:FIELd command selects which TV signal field is used during standard TV trigger mode. The line within the selected field is specified using the TRIGger:STV:LINE command.

Query

```
:TRIGger:STV:FIELd?
```

The query returns the current television signal field.

Returned Format

```
[:TRIGger:STV:FIELd] {1 | 2}
```

STV:LINE

Command

```
:TRIGger:STV:LINE <line_number>
```

The TRIGger:STV:LINE command selects the horizontal line in which the instrument will trigger on. Allowable line_number entry depends on the TRIGger:STV:FIELd and TRIGger:STV:STANdard selected. Once the vertical sync pulse of the selected field is received, the trigger is delayed by the number of lines specified.

`<line_number>` Is the horizontal line number. Allowable values range from 1 to 875 depending on TRIGger:STV:FIELd and TRIGger:STV:STANdard settings.

Query

```
:TRIGger:STV:LINE?
```

The query returns the current line number.

Returned Format

```
[:TRIGger:STV:LINE] <line_number>
```

STV:SOURce

Command

```
:TRIGger:STV:SOURce {CHANnelN | TRIGgerN}
```

The TRIGger:STV:SOURce command selects the source for standard TV mode triggering. This is the source that will be used for subsequent TRIGger:STV commands and queries.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` TRIGgerN, an integer 1 to 4, represents the slot where the external trigger resides.

Query

```
:TRIGger:STV:SOURce?
```

The query returns the currently selected standard TV trigger mode source.

Returned Format

```
[:TRIGger:STV:SOURce] {CHANnelN | TRIGgerN}
```

STV:SPOLarity

Command

```
:TRIGger:STV:SPOLarity {POSitive | NEGative}
```

The TRIGger:STV:SPOLarity (Sync POLarity) command specifies the vertical sync pulse polarity for the selected field used during standard TV mode triggering.

Query

```
:TRIGger:STV:SPOLarity?
```

The query returns the currently selected sync pulse polarity.

Returned Format

```
[:TRIGger:STV:SPOLarity] {POSitive | NEGative}
```

STV:STANdard

Command

```
:TRIGger:STV:STANdard {525 | 625 | 875}
```

The TRIGger:STV:STANdard command selects the television signal standard to be used during standard TV mode triggering.

- 525 selects 525 lines per frame at 60 frames per second.
- 625 selects 625 lines per frame at 50 frames per second.
- 875 selects 875 lines per frame at 60 frames per second.

Query

```
:TRIGger:STV:STANdard?
```

The query returns the currently selected television signal standard.

Returned Format

```
[:TRIGger:STV:STANdard] {525 | 625 | 875}
```

SWEep

Command

```
:TRIGger:SWEep {AUTO|TRIGgered}
```

The TRIGger:SWEep command selects the sweep mode. If the AUTO mode is selected and no trigger event occurs within 30 ms of enabling the trigger, the instrument automatically causes a sweep to occur by forcing a trigger. This process repeats.

If the TRIGGERED mode is selected, and no trigger is present, the unit will not sweep, and the data acquired on the previous trigger will remain on-screen.

Query

```
:TRIGger:SWEep?
```

The query returns the current sweep mode.

Returned Format

```
[:TRIGger:SWEep]        {AUTO|TRIGgered}
```

UDTV

Command

`:TRIGger:UDTV`

This command sets the conditions for the TV trigger mode using user-defined parameters. This mode is used for triggering on non-standard television signals, and allows you to define the conditions that must be met before a trigger occurs.

The :TRIGger:MODE command is used to select the user-defined TV trigger mode.

The :TRIGger:UDTV command is sent with one or more of the following commands and queries:

- ENUMber,
- PGTHan,
- PLTHan,
- SLOPe,
- SOURce, and
- STATe.

When triggering for user-defined TV mode:

- The channel or trigger source used for the trigger is set using the :TRIGger:UDTV:SOURce command.
- The conditions for arming the trigger are specified using the TRIGger:UDTV:PGTHan, TRIGger:UDTV:PLTHan, and TRIGger:UDTV:STATe commands.
- The delay after the trigger is armed is set using the :TRIGger:UDTV:ENUMber command.
- The signal edge that causes the trigger to occur after arming and delay is set using the :TRIGger:UDTV:SLOPe command.

UDTV:ENUMber

Command

```
:TRIGger:UDTV:ENUMber <count>
```

The :TRIGger:UDTV:ENUMber command specifies the number of events (horizontal sync pulses) to delay after arming the trigger before actually looking for the trigger event. Conditions for arming the trigger are specified using the TRIGger:UDTV:PGTHan, TRIGger:UDTV:PLTHan, and TRIGger:UDTV:STATe commands.

`<count>` Is the number of events to delay. Allowable values range from 1 to 16,000,000.

Query

```
:TRIGger:UDTV:ENUMber?
```

The query returns the currently programmed count value.

Returned Format

```
[:TRIGger:UDTV:ENUMber] <count>
```

UDTV:PGTHan

Command

```
:TRIGger:UDTV:PGTHan <lower_limit>
```

The :TRIGger:UDTV:PGTHan (Present Greater THan) command specifies the minimum pulse width of the signal used to arm the trigger used during user-defined trigger mode.

If :TRIGger:UDTV:PGTHan is greater than :TRIGger:UDTV:PLTHan, an error is placed in the error queue.

`<lower_limit>` Is the minimum pulse width from 3 ns to 160 ms.

Query

```
:TRIGger:UDTV:PGTHan?
```

The query returns the currently selected minimum pulse width.

Returned Format

```
[:TRIGger:UDTV:PGTHan] <lower_limit>
```

UDTV:PLTHan

Command

```
:TRIGger:UDTV:PLTHan <upper_limit>
```

The :TRIGger:UDTV:PLTHan (Present Less THan) command specifies the maximum pulse width of the signal used to arm the trigger used during user-defined trigger mode.

If :TRIGger:UDTV:PLTHan is less than :TRIGger:UDTV:PGTHan, an error is placed in the error queue.

`<upper_limit>` Is the maximum pulse width from 3 ns to 160 ms.

Query

```
:TRIGger:UDTV:PLTHan?
```

The query returns the currently selected maximum pulse width.

Returned Format

```
[:TRIGger:UDTV:PLTHan] <upper_limit>
```

UDTV:SLOPe

Command

```
:TRIGger:UDTV:SLOPe {POSitive | NEGative}
```

The TRIGger:UDTV:SLOPe command specifies the signal edge that causes a trigger to occur after arming and delay conditions have been satisified.

Query

```
:TRIGger:UDTV:SLOPe?
```

The query returns the currently selected slope parameter.

Returned Format

```
[:TRIGger:UDTV:SLOPe] {POSitive | NEGative}
```

UDTV:SOURce

Command

```
:TRIGger:UDTV:SOURce {CHANnelN | TRIGgerN}
```

The TRIGger:UDTV:SOURce command selects the source for user-defined TV mode triggering. This is the source that will be used for subsequent TRIGger:UDTV commands and queries.

`CHANnelN` N represents an integer, 1 to 4, followed by an optional letter, A or B (CHAN1 = CHAN1A). The integer is the slot in which the channel resides. The letter identifies which of two possible channels in the slot is being referenced.

`TRIGgerN` N, an integer 1 to 4, represents the slot where the external trigger resides.

Query

```
:TRIGger:UDTV:SOURce?
```

The query returns the currently selected user-defined TV trigger mode source.

Returned Format

```
[:TRIGger:UDTV:SOURce] {CHANnelN | TRIGgerN}
```

UDTV:STATe

Command

```
:TRIGger:UDTV:STATe {HIGH | LOW}
```

The :TRIGger:UDTV:STATe command specifies whether the signal used to arm the trigger must occur above (HIGH) or below (LOW) the trigger level. Trigger level is specified using the TRIGger:LEVel command.

Query

```
:TRIGger:UDTV:STATe?
```

The query returns the currently selected arming state.

Returned Format

```
[:TRIGger:UDTV:STATe] {HIGH | LOW}
```

23

TriggerN Commands

TriggerN Commands

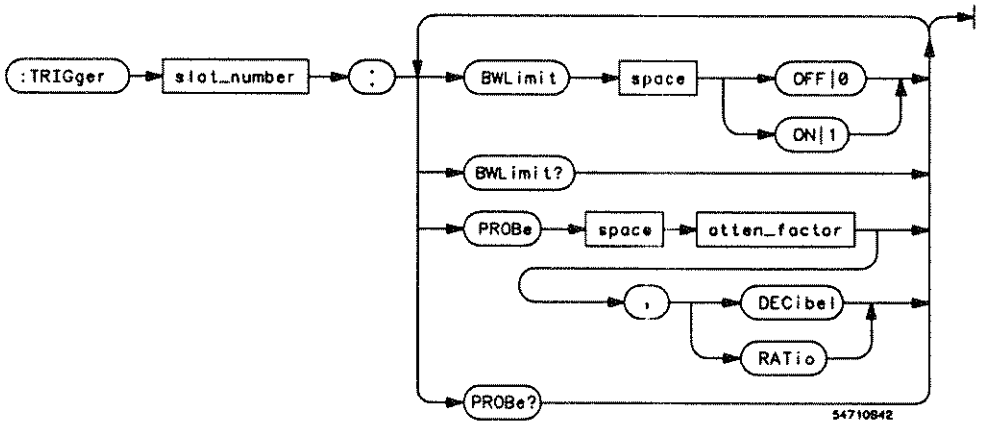
The TriggerN Subsystem contains commands and queries specific to a channel plug-in with external triggering capabilities. The HP 54711A, HP 54721A, and HP 54722A plug-ins have external triggering capabilities. These commands affect only the TriggerN input. The N at the end of trigger is the slot, 1 through 4, in which the external trigger input resides.

The TriggerN Subsystem consists of the following commands and queries:

- BWLimit, and
- PROBe.

Figure 23-1 is the syntax drawing of the TriggerN Subsystem.

Figure 23-1



TriggerN Subsystem Syntax Diagram

BWLimit

Command

```
:TRIGgerN:BWLimit {{ON | 1} | {OFF | 0}}
```

This command controls an internal lowpass filter in the HP 54711A plug-in and in TRIGger 1 of the HP 54722A plug-in. When ON, the bandwidth of the specified trigger channel is limited to approximately 100 MHz.

Example

The following example turns on the bandwidth limit filter for an HP 54711A plug-in in slot 1:

```
10  OUTPUT 707;":TRIGGER1:BWLIMIT ON"
20  END
```

Query

```
:TRIGgerN:BWLimit?
```

The query returns the current setting for the specified trigger input.

Returned Format

```
[:TRIGgerN:BWLimit]    {1|0}
```

Example

The following example gets the setting of the bandwidth limit filter for an HP 54711A plug-in in slot 1 and prints it on the controller screen:

```
10  OUTPUT 707;":TRIGGER1:BWLIMIT?"
20  ENTER 707;Bwlimit
30  PRINT Bwlimit
40  END
```

PROBe

Command

```
TRIGgerN:PROBe <attn_factor>, {DECibel | RATio}
```

This command sets the probe attenuation factors and units for the selected external trigger.

The following example sets the external trigger probe attenuation factor to 10:1 in slot 1:

```
10  OUTPUT 707;":TRIGGER1:PROBE 10,RATIO"
20  END
```

Query

```
:TRIGgerN:PROBe?
```

The query returns the current probe attenuation factor and the currently selected units for the external trigger.

Returned Format

```
[:TRIGgerN:PROBe]<attn_factor>, {DECibel | RATio}
```

Example

The following example gets the current attenuation factor for an external trigger probe attached to the plug-in in slot 1, and prints the factor on the controller screen:

```
10  DIM Factor$[15]
20  OUTPUT 707;":TRIGGER1:PROBE?"
30  ENTER 707;Factor$
40  PRINT Factor$
50  END
```

24

Waveform Commands

Waveform Commands

The WAVEFORM subsystem is used to transfer waveform data between a controller and the oscilloscope. It contains commands to set up the waveform transfer and to send or receive waveform records to or from the instrument. The waveform record is contained in two portions: the preamble and the waveform data. The preamble contains the scaling and other values used to describe the data. The waveform data contains the actual data in the waveform. The preamble and waveform data must be read or sent with two separate commands: WAVeform:PREamble and WAVeform:DATA.

The Waveform subsystem contains the following commands and queries:

- BANDpass?,
- BYTeorder,
- COMPlete?,
- COUNt?,
- COUPling?,
- DATA,
- FORMat,
- POINts?,
- PREamble,
- SOURce,
- TYPE?,
- VIEW,
- XDISplay?,
- XINCrement?,
- XORigin?,
- XRANge?,
- XREFerence?,
- XUNits?,
- YDISplay?,
- YINcrement?,
- YORigin?,
- YRANge?,
- YREFerence?, and
- YUNits?.

Figure 24-1 is the syntax diagram for the Waveform subsystem commands.

Data Acquisition

When the data is acquired using the Digitize command, the data is placed in the channel or function buffer of the specified source. After the Digitize command, the oscilloscope is stopped. If the oscilloscope is restarted over HP-IB or the front panel, the data acquired with the digitize command is overwritten.

The preamble, elements of the preamble, or waveform data can be queried while the oscilloscope is running, but the data will reflect only the current acquisition and subsequent queries will not reflect consistent data. For example, if the oscilloscope is running and the X origin is queried, then the data is queried in a separate HP-IB command, it is likely that the first point in the data will have a different time than that of the X origin. This is due to data acquisitions that may have occurred between the queries. For this reason, this mode of operation is not recommended. Instead, the Digitize command should be used to stop the oscilloscope so that all subsequent queries will be consistent.

The data in the channel, function, and memory buffers is non-volatile. Therefore, if the oscilloscope power is cycled, the acquired data may still be queried with the waveform query commands. However, it is not recommended.

Function data is volatile and must be read following a Digitize command or the data will be lost when the oscilloscope is turned off.

Waveform Data and Preamble

The waveform record is actually contained in two portions: the waveform data and the preamble. The waveform data is the actual data acquired for each point in the specified source. The preamble contains the information for interpreting the waveform data, which includes the number of points acquired, the format of the acquired data, and the type of acquired data. The preamble also contains the X and Y increments, origins, and references for the acquired data.

The values set in the preamble are determined when the DIGITIZE command is executed or when the front-panel STORE key is pressed. The preamble values are based on the settings of variables in the

ACQUIRE subsystem, or they are based on the front-panel setup when the STORE key is pressed.

Although preamble values can be changed with a controller, the way the data is acquired cannot be changed. Changing the preamble values cannot change the type of data that was actually acquired, the number of points actually acquired, etc. Therefore, you must use extreme caution when changing any waveform preamble values to ensure the data is still useful. For example, setting points in the preamble to a different value from the actual number of points in the waveform results in inaccurate data.

The waveform data and preamble must be read or sent with two separate commands: WAVeform:DATA and WAVeform:PREamble.

Data Acquisition Types

There are four types of data acquisition that can be selected with the ACQUIRE:TYPE command: normal, raw, interpolate, and average. Refer to the ACQUIRE:TYPE commands chapter for more information on data acquisition.

Data Conversion

Data sent from the oscilloscope must be scaled for useful interpretation. The values used to interpret the data are the X and Y origins, the X and Y increments, and the Y and Y references. These values can be read from the waveform preamble.

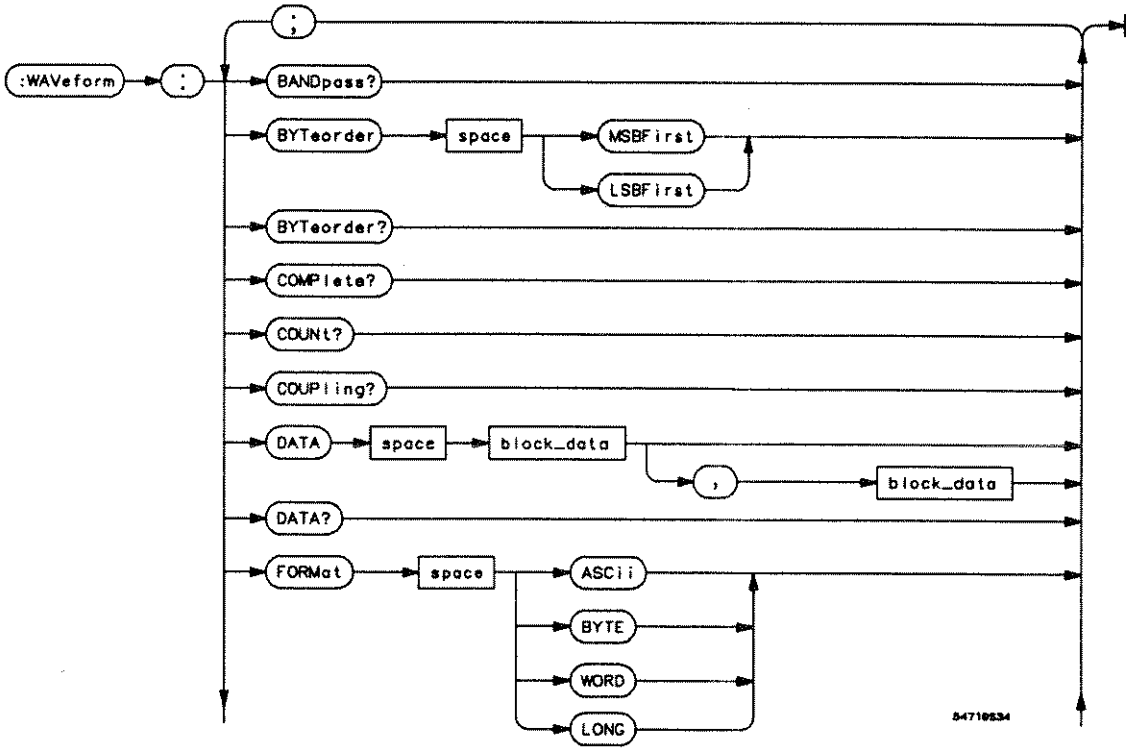
Conversion from Data Value to Units

$$Y\text{-}axis\ Units = (data\ value - Yreference) \times Yincrement + Yorigin$$
$$X\text{-}axis\ Units = (data\ value - Xreference) \times Xincrement + Xorigin$$

Data Format for HP-IB Transfer

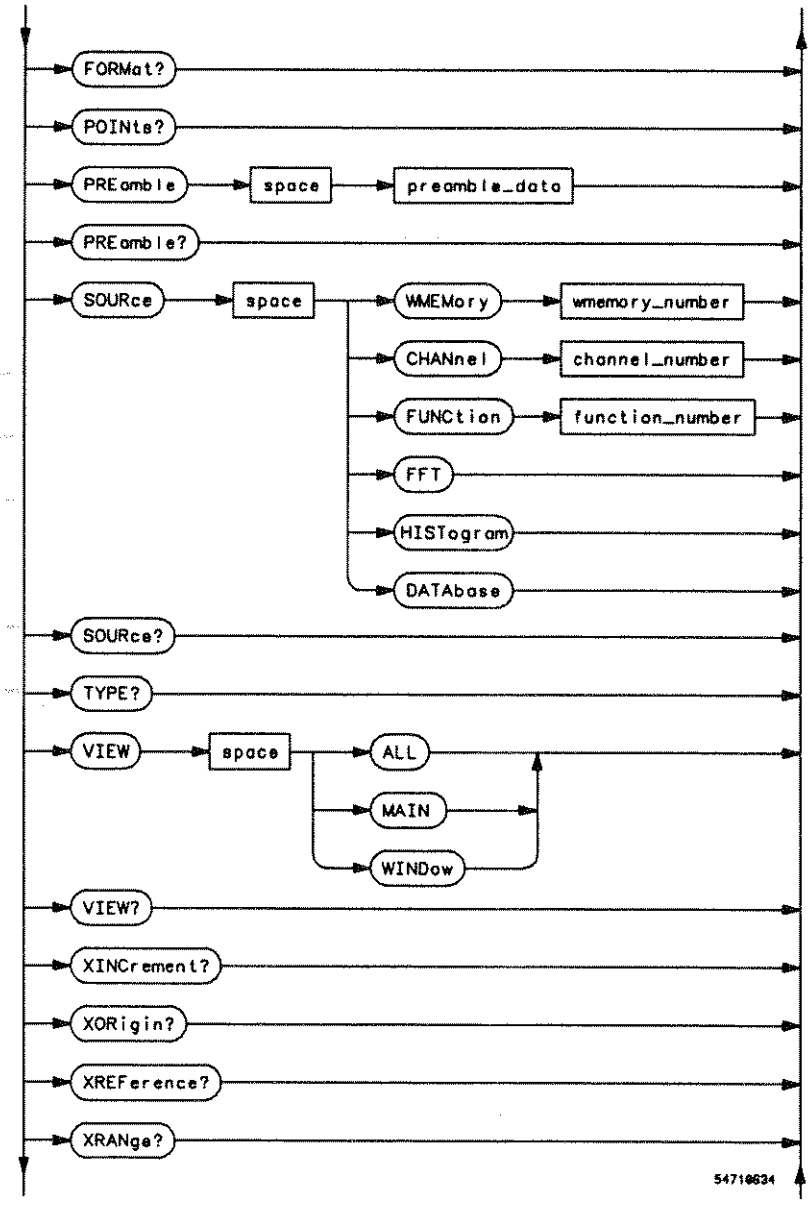
There are four types of data formats that can be selected with the :WAVEform:FORMat command: ASCII, byte, word, and long. Refer to the FORMat command for more information on data format.

Figure 24-1



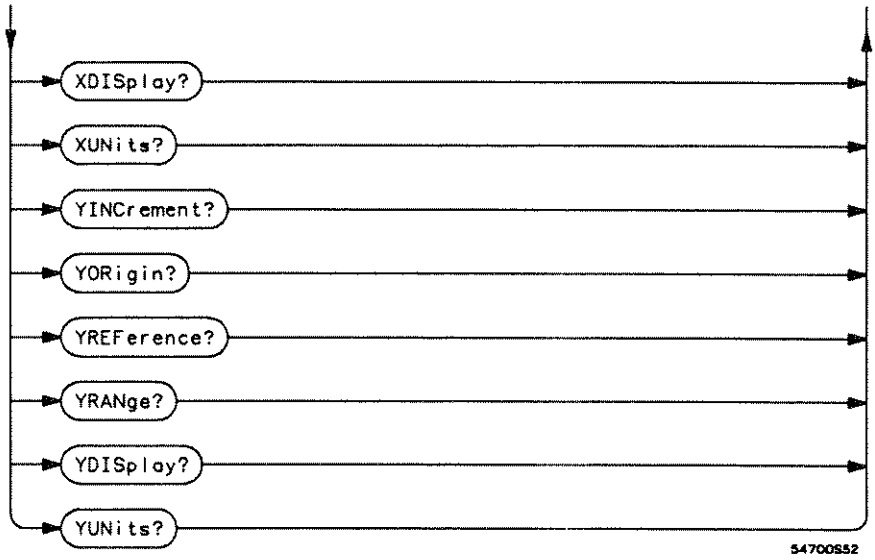
Waveform Subsystem Commands Syntax Diagram

Figure 24-1 (continued)



Waveform Subsystem Syntax Diagram (continued)

Figure 24-1 (continued)



block_data	block data in IEEE 488.2 # format.
preamble_data	refer to the PREamble command.
channel_number	an integer, 1 through 4.
function_number	an integer 1 or 2.
wmemory_number	an integer, 1 through 4.

Waveform Subsystem Syntax Diagram (continued)

BANDpass?

Query

```
:WAVeform:BANDpass?
```

The WAVeform:BANDpass query returns an estimation of the maximum and minimum bandwidth limits of the source signal. Bandwidth limits are computed as a function of the coupling and filter mode selected. Cutoff frequencies are derived from the acquisition path and software filtering.

Returned Format

```
[:WAVeform:BANDpass]<lower_cutoff>,<upper_cutoff><NL>
```

`<lower_cutoff>` minimum frequency passed by the acquisition

`<upper_cutoff>` maximum frequency passed by the acquisition

Example

The following example places the estimated maximum and minimum bandwidth limits of the source signal in the string variable "Bandpass$," then prints the contents of the variable to the controller's screen.

```
10  DIM Bandpass$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:BANDPASS?"
30  ENTER 707;Bandpass$
40  PRINT Bandpass$
50  END
```

BYTeorder

Command

```
:WAVeform:BYTeorder {MSBFirst | LSBFirst}
```

The WAVeform:BYTeorder command selects the order in which bytes are transferred to and from the oscilloscope over the HP-IB for the WORD and LONG formats. If MSBFirst is selected, the most significant byte is transferred first. Otherwise, the least significant byte is transferred first. The default is MSBFirst.

Example

The following example sets up the oscilloscope to send the most significant byte first during data transmission.

```
10  OUTPUT 707;":WAVEFORM:BYTEORDER MSBFIRST"
20  END
```

Query

```
:WAVeform:BYTeorder?
```

The WAVeform:BYTeorder query returns the current setting for the byte order.

Returned Format

```
[:WAVeform:BYTeorder] {MSBFirst | LSBFirst}<NL>
```

Example

The following example places the current setting for the byte order in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[10]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:BYTEORDER?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

MSBFirst is for microprocessors, like Motorola, where the most significant byte resides at the lower address. LSBFirst is for microprocessors, like Intel, where the least significant byte resides at the lower address.

COMPlete?

Query

```
:WAVeform:COMPlete?
```

The WAVeform:COMPlete query returns the percent of time buckets that are complete for the currently selected waveform.

For the NORMAL, RAW, and INTERPOLATE waveform types, the percent complete is the percent of the number of time buckets that have data in them compared to the record length.

For the AVERAGE waveform type, the percent complete is the percent of the number of time buckets that have the number of specified hits compared to the record length. The hits are specified by the WAVEFORM:COUNT command.

For the VERSUS waveform type, the percent complete is the least complete of the x-axis and y-axis waveforms.

Returned Format

```
[:WAVeform:COMPlete] <criteria><NL>
```

`<criteria>` 0 to 100 percent, rounded down to the closest integer.

Example

The following example places the current completion criteria in the string variable, Criteria$, then prints the contents of the variable to the controller's screen.

```
10  DIM Criteria$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:COMPLETE?"
30  ENTER 707;Criteria$
40  PRINT Criteria$
50  END
```

COUNt?

Query

```
:WAVeform:COUNt?
```

The WAVeform:COUNt query returns the fewest number of hits in all of the time buckets for the currently selected waveform. For the AVERAGE waveform type, the count value is the fewest number of hits for all time buckets. This value may be less than or equal to the value specified with the ACQuire:COUNt command.

For the NORMAL, RAW, INTERPOLATE, and VERSUS waveform types, the count value returned is one, unless the data contains holes. If the data contains holes, zero is returned.

Returned Format

```
[:WAVeform:COUNt] <number><NL>
```

`<number>` An integer. Values range from 1 to 262144 for NORMal, RAW, or INTerpolate types and from 1 to 32768 for VERSus type.

Example

The following example places the current value in the count field in the string variable, Count$, then prints the contents of the variable to the controller's screen.

```
10  DIM Count$[50]                    !Dimension variable
20  OUTPUT 707;":WAVEFORM:COUNT?"
30  ENTER 707;Count$
40  PRINT Count$
50  END
```

COUPling?

Query

```
:WAVeform:COUPling?
```

The WAVeform:COUPling query returns the input coupling of the currently selected waveform.

Returned Format

```
[:WAVeform:COUPling] {AC | DC | DCFifty | LFReject}<NL>
```

Example

The following example places the current input coupling of the selected waveform in the string variable, Setting$, then prints the contents of the variable.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:COUPLING?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

See Also

The CHANnel<number>:INPut command sets the coupling for a particular channel.

DATA

Command

```
:WAVeform:DATA <block_data>[,<block_data>]
```

The WAVeform:DATA command transfers waveform data to the oscilloscope over HP-IB and stores the data in a previously specified waveform memory. The waveform memory is specified with the WAVeform:SOURce command. Only waveform memories may have waveform data sent to them. The format of the data being sent must match the format previously specified by the waveform preamble for the destination memory.

VERSus data is transferred as two arrays. The first array contains the data on the x-axis and the second array contains the data on the y-axis. The two arrays are transferred one at a time over HP-IB in a linear format. There are n data points sent in each array where n is the number in the points portion of the preamble.

Database data is transferred as a two-dimensional array of unsigned word values from 0 to 63488. The two-dimensional array has dimensions of 256 rows by 451 columns and is row priority.

The full qlevel range of the A/D converter will be returned with the data query. The Y increment, Y origin, and Y reference values should be used to convert the qlevels to voltage values. The Y range and Y display values should be used to plot the voltage values. All of these reference values are available from the waveform preamble.

`<block_data>` Binary block data in the # format.

Example

The following example sends 1000 bytes of previously saved data to the oscilloscope from the array, Set.

```
10  OUTPUT 707 USING "#,K";:WAVEFORM:DATA #800001000"
20  OUTPUT 707 USING "W";Set(*)
30  END
```

is an HP BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item.

K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trauling blanks.

W is an HP BASIC image specifier that outputs 16-bit words with the most significant byte first.

Query

```
:WAVeform:DATA?
```

The WAVeform:DATA query outputs waveform data to the controller over HP-IB that is stored in a waveform memory, function, or channel buffer previously specified with the WAVeform:SOURce command. The data that is returned is described by the waveform preamble.

Returned Format

```
[:WAVeform:DATA] <block_data>[,<block_data>]<NL>
```

Example

The following example places the current waveform data from channel 1 of the array Wdata in the word format.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1    !Select source
30  OUTPUT 707;":WAVEFORM:FORMAT WORD"       !Select
40  OUTPUT 707;":WAVEFORM:DATA?"
50  ENTER 707 USING "#,2A,6D";Header$,Length
60  Length = Length/2                                !Length in words
70  ALLOCATE INTEGER Wdata(1:Length)
80  ENTER 707 USING "#,W";Wdata(*)
90  ENTER 707 USING "-K,B";End$
100 END
```

 # is an HP BASIC image specifier that terminates the statement when the last ENTER item is terminated. EOI and line feed are the item terminators.

2A is an HP BASIC image specifier that places the next two characters received in a string variable.

8D is an HP BASIC image specifier that places the next 8 characters in a numeric variable.

W is an HP BASIC image specifier that places the data in the array in word format with the first byte entered as the most significant byte.

-K is an HP BASIC image specifier that places the block data in a string, including carriage returns and line feeds until EOI is true or when the dimensioned length of the string is reached.

B is an HP BASIC specifier that enters the next byte in a variable.

The format of the waveform data must match the format previously specified by the WAVeform:FORMat, WAVeform:BYTeorder, and WAVeform:PREamble commands.

FORMat

Command

```
:WAVeform:FORMat {ASCii | BYTE | LONG | WORD}
```

The WAVeform:FORMat command sets the data transmission mode for waveform data output. This command controls how the data is formatted when the data is sent from the oscilloscope and pertains to all waveforms.

ASCII ASCII formatted data consists of ASCII digits with each data value separated by a comma. The data values can be converted to real values on the y-axis, ie. volts, and transmitted in floating point engineering notation.

In ASCII, the value "99.999E+36" represents a hole level (a hole in the acquisition data). The value "99.999E+33" represents a clipped-high level. The value "99.999E+30" represents a clipped-low level.

BYTE BYTE formatted data is formatted as signed 8-bit integers. If you use BASIC, you need to create a function to convert these signed bits to signed integers. In the byte format, the value 125 represents a hole level (a hole in the acquisition data). The value 127 represents a clipped-high level. The value 126 represents a clipped-low level. Data is rounded when converted from larger to a smaller size. For waveform transfer into the instrument, the maximum valid qlevel is 124. The minimum valid qlevel is − 128.

WORD WORD formatted data is transferred as signed 16-bit integers in two bytes. If WAVEFORM:BYTeorder is set to MSBFirst, then the most significant byte of each word is sent first. If the BYTeorder is LSBFirst, then the least significant byte of each word is sent first. In the word format, the value 31232 represents a hole level (a hole in the data acquisition). The value 32256 represents a clipped-high level. The value 31744 represents a clipped-low level. For waveform transfer into the instrument, the maximum valid qlevel is 30720. The minimum qlevel is - 32736.

The word format is also used to transfer database data to and from the instrument. Database data values will range from 0 through 63488 in an unsigned format.

LONG LONG formatted data is transferred as signed 32-bit integers in four bytes. If WAVeform:BYTeorder is set to MSBFirst, then the most significant byte of each long word is sent first. If the BYTeorder is LSBFirst, then the least significant byte of each long word is sent first. In the long format, the value 2046820352 represents a hole level (a hole in the data acquisition). The value 2113929216 represents a clipped-high level. The value 2080374784 represents a clipped-low level. For waveform transfer into the instrument, the maximum valid qlevel is 2013265920. The minimum qlevel is 0. The long format is used to transfer histogram data to and from the instument.

The default format is ASCII.

Example

The following example selects the WORD format for waveform data transmission.

```
10  OUTPUT 707;":WAVEFORM:FORMAT WORD"
20  END
```

Query

```
:WAVeform:FORMat?
```

The WAVeform:FORMat query returns the current output format for transferring waveform data.

Returned Format

```
[:WAVeform:FORMat] {ASCii | BYTE | LONG | WORD}<NL>
```

Example

The following example places the current output format for data transmission in the string variable, Mode$, then prints the contents of the variable to the controller's screen.

```
10  DIM Mode$[50]              !Dimension variable
20  OUTPUT 707;":WAVEFORM:FORMAT?"
30  ENTER 707;Mode$
40  PRINT Mode$
50  END
```

POINts?

Query

```
:WAVeform:POINts?
```

The WAVeform:POINts query returns the points value in the current waveform preamble. The points value is the number of time buckets contained in the waveform selected with the WAVeform:SOURce command.

Returned Format

```
[:WAVeform:POINts] <points><NL>
```

<points> An integer. Values range from 1 to 262144. The actual range depends on the oscilloscope model and plug-in configuration because different plug-ins have different maximum record lengths. See the ACQuire:POINts command for more information.

The number of points for vertical histograms will not exceed 256. The number of points for horizontal histograms will not exceed 451. The number of points for database waveforms will always be 115,456.

Example

The following example places the current acquisition length in the numeric variable, Length, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:POINTS?"
30  ENTER 707;Length
40  PRINT Length
50  END
```

When you are receiving numeric data into numeric variables, the headers should be turned off. Otherwise, the headers may cause misinterpretation of returned data.

See Also

The ACQuire:POINts command in the Acquire Commands chapter.

PREamble

Command `:WAVeform:PREamble <preamble_block>`

The WAVeform:PREamble command sends a waveform preamble to the previously selected waveform memory in the oscilloscope. The preamble contains the scaling and other values used to describe the data. The waveform memory is specified with the WAVeform:SOURce command. Only waveform memories may have waveform data sent to them. Table 24-1 lists the elements in the preamble.

The preamble can be used to translate raw data into time and voltage values.

`<preamble_block>` <format NR1>, <type NR1>, <points NR1> ,<count NR1> ,
<X increment NR3> ,<X origin NR3>,< X reference NR1>,
<Y increment NR3>, <Y origin NR3> ,<Y reference NR1>,
<coupling NR1>,
<X display range NR3>, <X display origin NR3>,
<Y display range NR3>, <Y display origin NR3>,
<date, string>, <time, string>,
<frame model #, string>, <plug-in model #, string>,
<acquisition mode NR1>, <completion NR1>,
<X units NR1>, <Y units NR1>,
<max bandwidth limit NR3>, <min bandwidth limit NR3>

`<date>` A string containing the data in the format DD MMM YYYY where DD is the day, 1 to 31; MMM is the month; and YYYY is the year.

`<time>` A string containing the time in the format HH:MM:SS:TT where HH is the hour, 0 to 23, MM is the minute, 0 to 59, SS is the second, 0 to 59, and TT is the hundreds of seconds, 0 to 99.

`<frammodel #>` A string containing the model number and serial number of the frame in the format MODEL#:SERIAL#.

`<plug-in model #>` A string containing the model number and serial number of the plug-in in the format MODEL#:SERIAL#. These values are zero if no plug-in is used.

`<format>` 0 for ASCII format.
1 for BYTE format.
2 for WORD format.
3 for LONG format.

`<type>` 1 for RAW type.
2 for AVERAGE type.
3 for VHISTOGRAM (vertical histogram) type.
4 for HHISTOGRAM (horizontal histogram) type.
5 for VERSUS type.
6 for INTERPOLATE type.
7 for NORMAL type.
8 for DATABASE type.

`<acquisition mode>` 0 for REALTIME mode.
1 for EQUIVALENT TIME mode.

`<coupling>` 0 for AC coupling.
1 for DC coupling.
2 for DCFIFTY coupling.
3 for LFREJECT coupling.

`<x units>` 0 for UNKNOWN units.
`<y units>` 1 for VOLT units.
2 for SECOND units.
3 for CONSTANT units.
4 for AMP units.
5 for DECIBEL units.
6 for HITS unit.

See Table 24-1 for descriptions of all the waveform preamble elements.

 # is an HP BASIC image specifier that suppresses the automatic output of the EOL sequence following the last output item.

K is an HP BASIC image specifier that outputs a number or string in standard form with no leading or trauling blanks.

In line 20 of the program example, a space is inserted between the word "PREAMBLE" and the closed quotation mark. This space must be inside the quotation mark because in this format (#,K), the data is packed together. Failure to add the space produces a word that is not a proper command.

Query

```
:WAVeform:PREamble?
```

The WAVeform:PREamble outputs a waveform preamble to the controller from the waveform source, which can be a waveform memory or channel buffer.

Returned Format

```
[:WAVeform:PREamble] <preamble_block><NL>
```

Example

The following example outputs the current waveform preamble for the selected source to the string variable, Preamble$.

```
10  DIM Preamble$[250]              !Dimension variable
20  OUTPUT 707;":SYSTEM:HEADER OFF" !Response headers off
30  OUTPUT 707;":WAVEFORM:PREAMBLE?"
40  ENTER 707 USING "-K";Preamble$
50  END
```

-K is an HP BASIC image specifier that places the block data in a string, including carriage returns and line feeds until EOI is true or when the dimensioned length of the string is reached.

Table 24-1 Waveform Preamble Elements

Element	Description
Format	The format value describes the data transmission mode for waveform data output. This command controls how the data is formatted when it is sent from the oscilloscope. (See WAVeform:FORMat)
Type	This value describes how the waveform was acquired. (See also: WAVeform:TYPE)
Points	The number of data points or data pairs contained in the waveform data. (See ACQuire:POINts.) The number of points for vertical histograms will not exceed 256. The number of points for horizontal histograms will not exceed 451. The number of points for database waveforms will always be 115,456.

Element	Description
Count	For the AVERAGE waveform type, the count value is the minimum count or fewest number of hits for all time buckets. This value may be less than or equal to the value requested with the ACQuire:COUNt command. For NORMAL, RAW, INTERPOLATE, and VERSUS waveform types, this value is 0 or 1. The count value is ignored when it is sent to the oscilloscope in the preamble. (See WAVeform:TYPE and ACQuire:COUNt)
X increment	The X increment is the duration between data points on the X-axis. For time domain signals, this is the time between points. (See WAVeform:XINCrement)
X Origin	The X origin is the xaxis value of the first data point in the data record. For time domain signals, it is the time of the first point. This value is treated as a double precision 64-bit floating point number. (See WAVeform:XORigin)
X Reference	The X reference is the data point associated with the X origin. It is at this data point that the X origin is defined. In this oscilloscope, the value is always zero. (See WAVeform:XREFerence)
Y Increment	The Y increment is the duration between y-axis levels. For voltage waveforms, it is the voltage corresponding to one level. (See WAVeform:YINCrement)
Y Origin	The Y origin is the y-axis value at level zero. For voltage signals, it is the voltage at level zero. (See WAVeform:YORigin)
Y Reference	The Y reference is the level associated with the Y origin. It is at this level that the Y origin is defined. In this oscilloscope, this value is always zero. (See WAVeform:YREFerence)
Coupling	The input coupling of the waveform. The coupling value is ignored when sent to the oscilloscope in the preamble. (See WAVeform:COUPling)
X Display Range	The X display range is the x-axis duration of the waveform that is displayed. For time domain signals, it is the duration of time across the display. (See WAVeform:XRANge)
X Display Origin	The X display origin is the x-axis value at the left edge of the display. For time domain signals, it is the time at the start of the display. This value is treated as a double precision 64-bit floating point number. (see WAVeform:XDISplay)
Y Display Range	The Y display range is the y-axis duration of the waveform which is displayed. For voltage waveforms, it is the amount of voltage across the display. (See WAVeform:YRANge)
Y Display Origin	The y-display origin is the y-axis value at the center of the display. For voltage signals, it is the voltage at the center of the display. (See WAVeform:YDISplay)
Date	The date that the waveform was acquired or created.
Time	The time that the waveform was acquired or created.

Element	Description
Frame Model #	The model number of the frame that acquired or created this waveform. The frame model number is ignored when it is sent to an oscilloscope in the preamble.
Plug-in Model #	The model number of the plug-in that acquired this waveform. If no plug-in was used, then the value in this field is zero. The plug-in model number is ignored when it is sent to the oscilloscope in the preamble.
Acquisition Mode	The acquisition sampling mode of the waveform. (See ACQuire:MODE)
Complete	The complete value is the percent of time buckets that are complete. The complete value is ignored when it is sent to the oscilloscope in the preamble. (See WAVeform:COMPlete)
X Units	The x-axis units of the waveform. (See WAVeform:XUNits)
Y Units	The y-axis units of the waveform. (See WAVeform:YUNits)
Band Pass	The band pass is two values that are an estimation of the maximum and minimum bandwidth limits of the source signal. Bandwidth limits are computed as a function of the coupling and filter mode selected. (See WAVeform:BANDpass)

SOURce

Command

```
:WAVeform:SOURce {WMEMory<number> |
FUNCtion<number> | CHANnel<number> | DATAbase |
HISTogram | FFT}
```

The WAVeform:SOURce command selects a channel, function, waveform memory, database, histogram, or FFT as the waveform source.

`<number>` For channels: the number represents an integer, 1 through 4

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following example selects channel 1 as the waveform source.

```
10  OUTPUT 707;":WAVEFORM:SOURCE CHANNEL1"
20  END
```

Query

```
:WAVeform:SOURce?
```

The WAVeform:SOURce query returns the currently selected waveform source.

Returned Format

```
[:WAVeform:SOURce] {WMEMory<number> | FUNCtion<number> |
CHANnel<number> | DATAbase | HISTogram | FFT}<NL>
```

Example

The following example places the current selection for the waveform source in the string variable, Selection$, then prints the contents of the variable to the controller's screen.

```
10  DIM Selection$[50]        !Dimension variable
20  OUTPUT 707;":WAVEFORM:SOURCE?"
30  ENTER 707;Selection$
40  PRINT Selection$
50  END
```

TYPE?

Query

```
:WAVeform:TYPE?
```

The WAVeform:TYPE query returns the current acquisition data type for the currently selected source. The type returned describes how the waveform was acquired. The waveform type may be NORMAL, RAW, INTERPOLATE, AVERAGE, DATABASE, HHISTOGRAM, VHISTOGRAM, or VERSUS.

NORMAL Normal data consists of the last data point in each time bucket.

RAW Raw data consists of one data point in each time bucket with no interpolation.

INTERPOLATE In the interpolate acquisition type the last data point in each time bucket is stored and additional data points are filled in between the acquired data points by interpolation.

AVERAGE Average data consists of the average of the first n hits in a time bucket, where n is the value in the count portion of the preamble. Time buckets that have fewer than n hits return the average of the data they contain. If the ACQuire:COMPlete parameter is set to 100%, then each time bucket must contain the number of data hits specified with the ACQuire:COUNt command.

DATABASE The database is transferred using unsigned values in the word format. The database is transferred as a block of data representing a two-dimensional array of 256 rows by 451 columns. The database may be generated using the DISPlay:CGRade command.

HHISTOGRAM The data is a horizontal histogram. Histograms are transferred using the LONG format. They can be generated using the HISTogram subsystem commands.

VHISTOGRAM The data is a vertical histogram. Histograms are transferred using the LONG format. They can be generated using the HISTogram subsystem commands.

VERSUS VERSus data consists of two arrays of data: one containing the x axis values, and the other containing the y axis values. Versus waveforms can be generated using the FUNCTION subsystem commands.

Returned Format

```
[:WAVeform:TYPE] {NORMal | RAW | INTerpolate | AVERage |
DATAbase | HHIStogram | VHIStogram | VERSus}<NL>
```

Example

The following example places the current acquisition data type in the string variable, Type$, then prints the contents of the variable to the controller's screen.

```
10  DIM Type$[50]              !Dimension variable
20  OUTPUT 707;":WAVEFORM:TYPE?"
30  ENTER 707;Type$
40  PRINT Type$
50  END
```

See Also

The acquisition data type is set with the ACQuire:TYPE command.

VIEW

Command

```
:WAVeform:VIEW {ALL | MAIN | WINDow}
```

The WAVeform:VIEW command selects which view of the waveform is selected for data and preamble queries. The command can be set to ALL, MAIN, or WINDOW. The view has different meanings depending upon the waveform source selected.

Channels

For channels, ALL, MAIN, or WINDOW views may be selected. If ALL is specified, all of the data in the waveform record is referenced. If MAIN is selected, only the data in the main time base range is referenced. The first value corresponds to the first time bucket in the main time base range and the last value corresponds to the last time bucket in the main time base range. If WINDOW is selected, only the data in the window time base range is referenced. The first value corresponds to the first time bucket in the window time base range and the last value corresponds to the last time bucket in the window time base range.

Memories

For memories, if ALL is specified, all the data in the waveform record is referenced. WINDOW and MAIN refer to the data contained in the memory time base range for the particular memory. The first value corresponds to the first time bucket in the memory time base range and the last value corresponds to the last time bucket in the memory time base range.

Functions

For functions, ALL, MAIN, and WINDOW refer to all of the data in the waveform record.

The default setting for this command is ALL. Table 24-2 summarizes the parameters for this command for each source.

Example

The following example sets up the oscilloscope to view all of the data.

```
10  OUTPUT 707;":WAVEFORM:VIEW ALL"
20  END
```

Table 24-2 **Waveform View Parameters**

Source/Parameter	ALL	MAIN	WINDOW
CHANNEL	All data	Main time base	Window time base
MEMORY	All data	Memory time base	Memory time base
FUNCTION	All data	All data	All data

Query

```
:WAVeform:VIEW?
```

The WAVeform:VIEW query returns the view of the data currently selected.

Returned Format

```
[:WAVeform:VIEW] {ALL | MAIN | WINDow}<NL>
```

Example

The following example returns the current view setting to the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10  DIM Setting$[50]          !Dimension variable
20  OUTPUT 707;":WAVEFORM:VIEW?"
30  ENTER 707;Setting$
40  PRINT Setting$
50  END
```

XDISplay?

Query

`:WAVeform:XDISplay?`

The WAVeform:XDISplay query returns the x-axis value at the left edge of the display. For time domain signals, it is the time at the start of the display. For VERSus type waveforms, it is the value at the center of the x-axis of the display. This value is treated as a double precision 64-bit floating point number.

Returned Format

`[:WAVeform:XDISplay] <value><NL>`

`<value>` A real number representing the x-axis value at the left edge of the display.

Example

The following example returns the x-axis value at the left edge of the display to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XDISPLAY?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

XINCrement?

Query `:WAVeform:XINCrement?`

The WAVeform:XINCrement query returns the duration between data points on the x-axis. For time domain signals, this is the time difference between consecutive data points for the currently specified waveform source. For VERSus type waveforms, this is the duration between levels on the x-axis. For voltage waveforms, this is the voltage corresponding to one level.

Returned Format `[:WAVeform:XINCrement] <value><NL>`

`<value>` A real number representing the duration between data points on the x-axis.

Example

The following example places the current Xincrement value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XINCREMENT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xincrement value can also be obtained through the WAVeform:PREamble query.

XORigin?

Query

```
:WAVeform:XORigin?
```

The WAVeform:XORigin query returns the x-axis value of the first data point in the data record. For time domain signals, it is the time of the first point. For VERSus type waveforms, it is the x-axis value at level zero. For voltage waveforms, it is the voltage at level zero. The value returned by this query is treated as a double precision 64-bit floating point number.

Returned Format

```
[:WAVeform:XORigin] <value><NL>
```

`<value>` A real number representing the x-axis value of the first data point in the data record.

Example

The following example places the current Xorigin value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XORIGIN?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xorigin value can also be obtained through the WAVeform:PREamble query.

XRANge?

Query `:WAVeform:XRANge?`

The WAVeform:XRANge query returns the x axis duration of the displayed waveform. For time domain signals, it is the duration of the time across the display. For VERSus type waveforms, it is the duration of the waveform that is displayed on the x axis.

Returned Format `[:WAVeform:XRANge] <value><NL>`

`<value>` A real number representing the x axis duration of the displayed waveform.

Example The following example returns the x axis duration of the displayed waveform to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XRANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

XREFerence?

Query

```
:WAVeform:XREFerence?
```

The WAVeform:XREFerence query returns the data point or level associated with the Xorigin data value. It is at this data point or level that the X origin is defined. In this oscilloscope, the value is always zero.

Returned Format

```
[:WAVeform:XREFerence] 0<NL>
```

Example

The following example places the current X Reference value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:XREFERENCE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Xreference value can also be obtained through the WAVeform:PREamble query.

XUNits?

Query

```
:WAVeform:XUNits?
```

The WAVeform:XUNits query returns the x-axis units of the currently selected waveform source. The currently selected source may be a channel, function, database, histogram, or waveform memory.

Returned Format

```
[:WAVeform:XUNits] {UNKNown | VOLT | SECond | CONStant | AMP |
DECibels | HITS}<NL>
```

Example

The following example returns the x-axis units of the currently selected waveform source to the string variable, Unit$, then prints the contents of the variable to the controller's screen.

```
10  DIM Unit$[50]              !Dimension variable
20  OUTPUT 707;":WAVEFORM:XUNITS?"
30  ENTER 707;Unit$
40  PRINT Unit$
50  END
```

YDISplay?

Query

```
:WAVeform:YDISplay?
```

The WAVeform:YDISplay query returns the y-axis value at the center of the display. For voltage signals, it is the voltage at the center of the display.

Returned Format

```
[:WAVeform:YDISplay] <value><NL>
```

<value> A real number representing the y-axis value at the center of the display.

Example

The following example returns the current Y Display value to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"        !Response headers off
20  OUTPUT 707;"":WAVEFORM:YDISPLAY?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YINCrement?

Query

`:WAVeform:YINCrement?`

The WAVeform:YINCrement query returns the duration between the y-axis levels.

For BYTE and WORD data, and for voltage waveforms, it is the voltage corresponding to one level.

For ASCII data format the YINCrement is the full voltage range covered by the A/D converter.

Returned Format

`[:WAVeform:YINCrement] <real_value><NL>`

`<real_value>` A real number in exponential (NR3) format.

Example

The following example places the current Yincrement value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YINCREMENT?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Yincrement value can also be obtained through the WAVeform:PREamble query.

YORigin?

Query

```
:WAVeform:YORigin?
```

The WAVeform:YORigin query returns the y-axis value at level zero.

For BYTE and WORD data and voltage signals, it is the voltage at level zero.

For ASCII data format, the YORigin is the y-axis value at the center of the data range. Data range is returned in the Y increment.

Returned Format

```
[:WAVeform:YORigin] <real_value><NL>
```

`<real_value>` A real number in exponential (NR3) format.

Example

The following example places the current Y origin value in the numeric variable, Center, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YORIGIN?"
30  ENTER 707;Center
40  PRINT Center
50  END
```

See Also

The YORIGIN value can also be obtained through the WAVeform:PREamble query.

YRANge?

Query `:WAVeform:YRANge?`

The WAVeform:YRANge query returns the y-axis duration of the displayed waveform. For voltage signals, it is the amount of voltage across the display.

Returned Format `[:WAVeform:YRANge] <value><NL>`

`<value>` A real number representing the y-axis duration of the displayed waveform.

Example

The following example returns the current Y Range value to the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"        !Response headers off
20  OUTPUT 707;":WAVEFORM:YRANGE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

YREFerence?

Query `:WAVeform:YREFerence?`

The WAVeform:YREFerence query returns the level associated with the Y origin. It is at this level that the Y origin is defined. In this oscilloscope, the value is always zero.

Returned Format `[:WAVeform:YREFerence]<integer_value><NL>`

`<integer_value>` Always 0.

Example

The following example places the current Y Reference value for the currently specified source in the numeric variable, Value, then prints the contents of the variable to the controller's screen.

```
10  OUTPUT 707;":SYSTEM:HEADER OFF"          !Response headers off
20  OUTPUT 707;":WAVEFORM:YREFERENCE?"
30  ENTER 707;Value
40  PRINT Value
50  END
```

See Also

The Yreference value can also be obtained through the WAVeform:PREamble query.

YUNits?

Query

```
:WAVeform:XUNits?
```

The WAVeform:YUNits query returns the y-axis units of the currently selected waveform source. The currently selected source may be a channel, function, database, histogram, or waveform memory.

Returned Format

```
[:WAVeform:YUNits] {UNKNown | VOLT | SECond | CONStant | AMP |
DECibels | HITS}<NL>
```

Example

The following example returns the y-axis units of the currently selected waveform source to the string variable, Unit$, then prints the contents of the variable to the controller's screen.

```
10  DIM Unit$[50]              !Dimension variable
20  OUTPUT 707;":WAVEFORM:YUNITS?"
30  ENTER 707;Unit$
40  PRINT Unit$
50  END
```

25

Waveform Memory Commands

Waveform Memory Commands

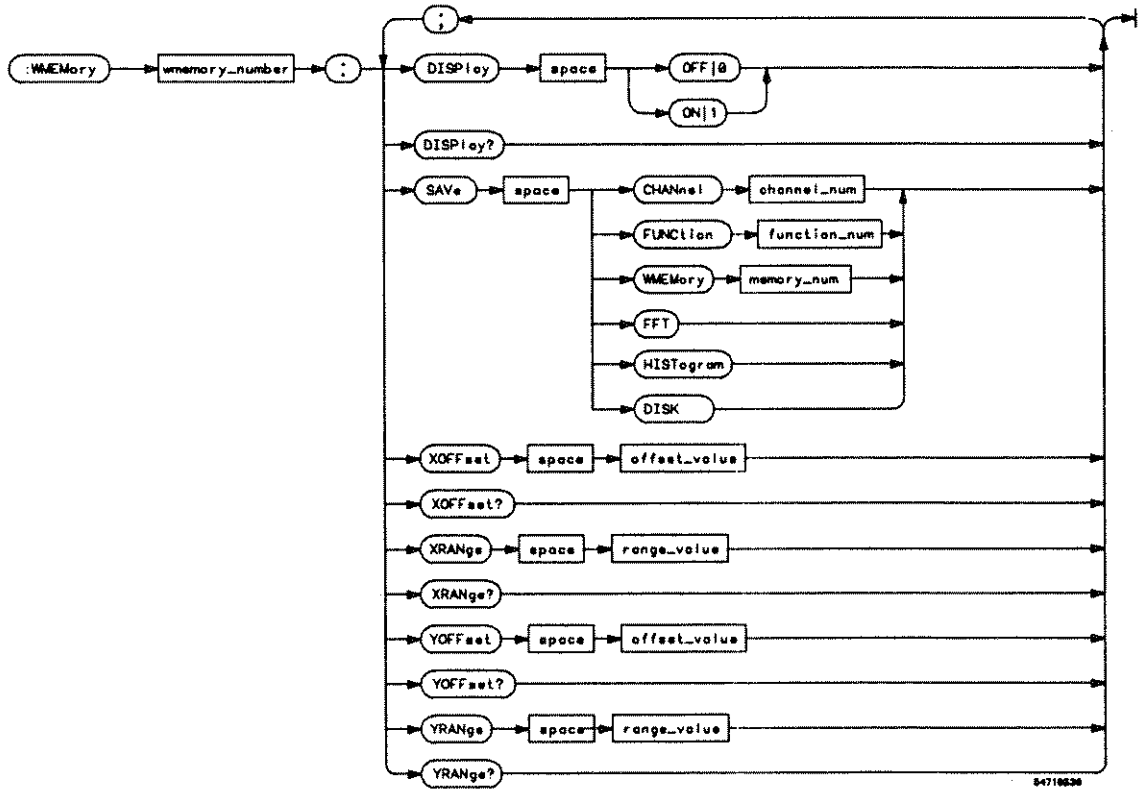
The Waveform Memory (:WMEMoryN) Subsystem commands control the waveform save functions. They allow saving and displaying of waveforms, memories, and functions.

The Waveform Memory Subsystem consists of the following commands and queries:

- DISPlay,
- SAVe,
- XOFFset,
- XRANge,
- YOFFset, and
- YRANge.

Figure 25-1 is the syntax diagram of the Waveform Memory Subsystem.

Figure 25-1



offset_value time on voltage of offset

range_value time on voltage of range

channel_num 1, 2, 3, or 4

function_num 1 or 2

memory_num 1, 2, 3, or 4

Waveform Memory Syntax Diagram

DISPlay

Command

```
:WMEMoryN:DISPlay {{ON | 1} | {OFF | 0 }}
```

This command enables or disables the viewing of the selected memory.

Query

```
:WMEMoryN:DISPlay?
```

The query returns the state of the selected memory.

Returned Format

```
[:WMEMoryN:DISPlay] {1 | 0} <NL>
```

SAVE

Command

```
:WMEMoryN:SAVE {CHANnelN | WMEMoryN | FUNCtionN | HISTogram | FFT}
```

The WMEMoryN:SAVE command stores the specified channel, waveform memory, function, histogram, or FFT waveform to the waveform memory.

See the User's Guide for information on how waveform memories are allocated for record lengths exceeding 64K points.

XOFFset

Command `:WMEMoryN:XOFFset <offset>`

The :WMEMoryN:XOFFset command sets the x-axis (horizontal) offset for the selected waveform memory's display scale. Offset is referenced to center screen.

`<offset>` Number that sets the offset value.

Query `:WMEMoryN:XOFFset?`

The query returns the current x-axis (horizontal) offset for the selected waveform memory.

Returned Format `[:WMEMoryN:XOFFset] <offset><NL>`

XRANge

Command `:WMEMoryN:XRANge <range>`

The :WMEMoryN:XRANge command sets the x-axis (horizontal) range (scale) for the selected waveform memory's display scale.

`<range>` Value that sets the range.

Query `:WMEMoryN:XRANge?`

The query returns the current x-axis (horizontal) range (scale) for the selected waveform memory.

Returned Format `[:WMEMoryN:XRANge] <range><NL>`

YOFFset

Command

```
:WMEMoryN:YOFFset< offset>
```

The :WMEMoryN:YOFFset command sets the vertical axis offset for the selected waveform memory.

`<offset>` Number that sets the offset value.

Query

```
:WMEMoryN:YOFFset?
```

The query returns the current offset for the selected waveform memory.

Returned Format

```
[:WMEMoryN:YOFFset]<offset><NL>
```

YRANge

Command

```
:WMEMoryN:YRANge <range>
```

The :WMEMoryN:YRANge command sets the vertical scaling for the selected memory.

`<range>` Number that sets the range value.

Query

```
:WMEMoryN:YRANge?
```

The query returns the Yaxis range for the selected memory.

Returned Format

```
[:WMEMoryN:YRANge]<range><NL>
```

26

FFT Commands

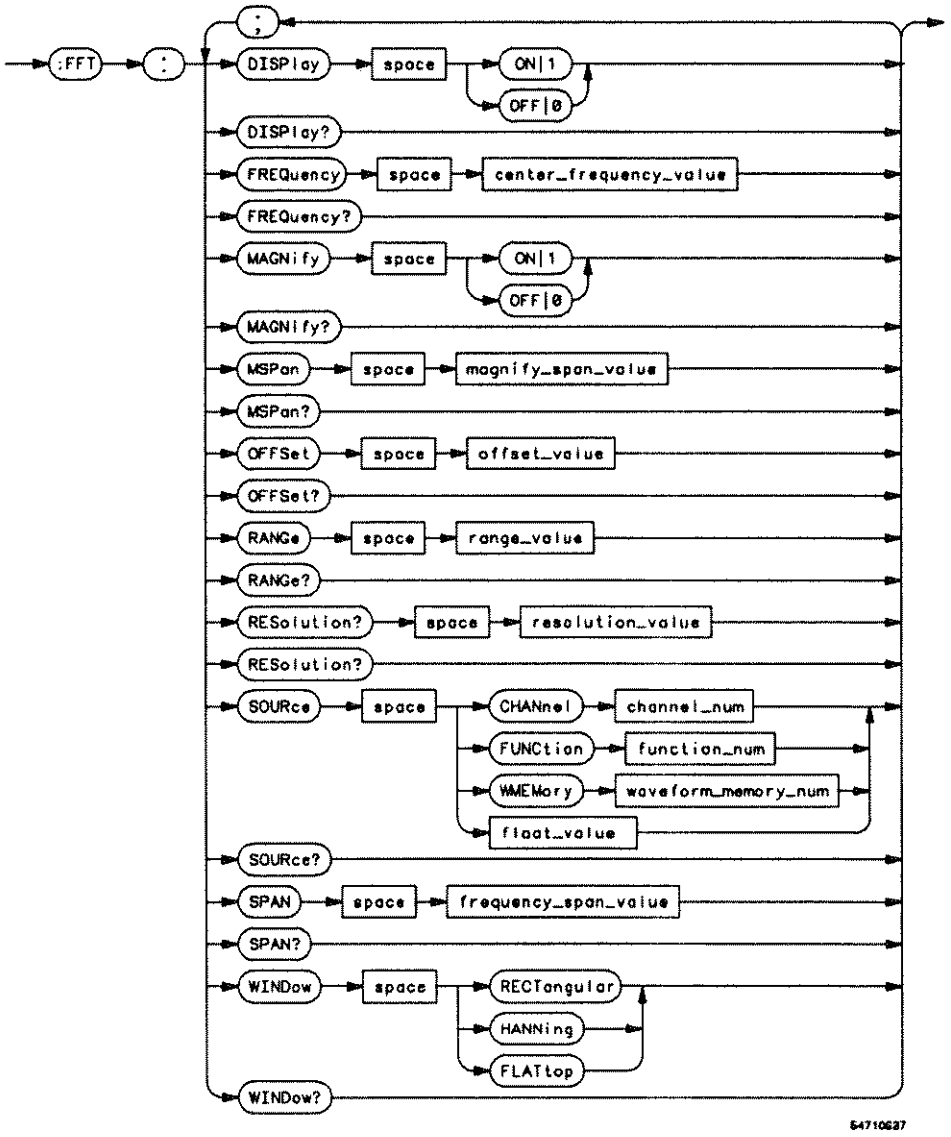
FFT Commands

The commands in this chapter turn on, scale, and set the FFT (fast Fourier transform) to particular windows. Other FFT commands are located in the FUNCtion and MEASure subsystems.

- DISPlay,
- FREQuency,
- MAGNify,
- MSPan,
- OFFSet,
- RANGe,
- RESolution,
- SOURce,
- SPAN, and
- WINDow.

Figure 26-1 is the FFT subsystem syntax diagram.

Figure 26-1



FFT Subsystem Syntax Diagram

DISplay

Command

```
FFT:DISPlay {{ON | 1} | {OFF | 0}}
```

The FFT:DISPlay command turns the FFT function on and off.

Example

The following example turns on the FFT display:

```
10  OUTPUT 707;":FFT:DISPLAY ON"
20  END
```

Query

```
FFT:DISPlay?
```

The query returns the current setting of the FFT display.

Returned Format

```
[FFT:DISPlay] {1 | 0}<NL>
```

FREquency

Command

```
FFT:FREQuency <frequency_value>
```

This command sets the center frequency value of the FFT display when MAGNify is on. Magnify is turned on and off with the FFT:MAGNify command.

`<frequency_value>` A frequency from 0 Hz to 1.0 times the unmagnified span.

Example

The following example sets the FFT center frequency value at 25 MHz.

```
10 OUTPUT 707;"FFT:FREQUENCY 25E6"
20 END
```

Query

```
FFT:FREQuency?
```

The query returns the current center frequency value of the FFT set with the command.

Returned Format

```
[FFT:FREQuency] <frequency_value><NL>
```

Example

The following example places the current setting of the center frequency value of the FFT in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:FREQUENCY?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

MAGNify

Command

```
FFT:MAGNify {{ON | 1 } | {OFF | 0}}
```

This command turns the magnification mode of the FFT on and off. When magnify is on, changing MSPAN and FREQuency results in software magnification of the display. The hardware setup is not changed.

Example

```
10 OUTPUT 707;"FFT:MAGNIFY ON"
20 END
```

Query

```
FFT:MAGNify?
```

The query returns the current setting of the magnification mode.

Returned Format

```
[FFT:MAGNify] { 1 | 0}<NL>
```

Example

The following example places the current setting of the magnify mode in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:MAGNIFY?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

MSPan

Command

```
FFT:MSPan <magnify_span_value>
```

This command sets the span when magnify is on. Magnify is controlled with the FFT:MAGNify command. MSPAN causes software magnification of the FFT display. The hardware setup is not affected.

`<magnify_span_value>` Span frequency in Hertz. Allowable values range from 1 to 1/200 Hz plus the value of (unmagnified) SPAN.

Example

The following example sets the FFT magnify span value to 25 MHz.

```
10 OUTPUT 707;":FFT:MSPAN 25E6"
20 END
```

Query

```
FFT:MSPan?
```

The query returns the current setting of the magnify span value.

Returned Format

```
[FFT:MSPan] <magnify_span_value> <NL>
```

Example

The following example places the current setting of the magnify span value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:MSPAN?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

OFFSet

Command

```
FFT:OFFSet <offset_value>
```

This command sets the dBm offset value represented at the center of the screen for the FFT display. This command controls software magnification of the FFT display. It does not affect the hardware settings.

<offset_value> Offset value in dBm.

Query

```
FFT:OFFSet?
```

The query returns the current FFT offset value.

Returned Format

```
[FFT:OFFSet] <offset_value><NL>
```

RANGe

Command

`FFT:RANGe <range_value>`

This command sets the vertical range value for the FFT. This command controls software scaling of the FFT display . The hardware settings are not affected.

`<range_value>` Full-scale vertical range in dB.

Query

`FFT:RANGe?`

The query returns the current vertical range setting for the FFT.

Returned Format

`[FFT:RANGe] <range_value><NL>`

Example

The following example places the current setting of the range value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:RANGE?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

RESolution

Command

```
FFT:RESolution <resolution_value]
```

This command sets the frequency resolution for the FFT. This command changes the waveform record length that is normally set in the acquisition menu or by the ACQuire:POINts command. The record length is set according to the equation

$$record\ length = \frac{sample\ rate}{resolution.}$$

<resolution_value> Spectral resolution in Hertz.

Query

```
FFT:RESolution?
```

The query returns the current resolution setting of the FFT.

Returned Format

```
[FFT:RESolution] <resolution_value><NL>
```

Example

The following example places the current setting of the resolution value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:RESOLUTION?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

SOURce

Command

```
FFT:SOURce {CHANnel<number> | FUNCtion<number> |
WMEMory<number> | <float_value>}
```

This command selects the source for the FFT.

`<number>` For channels: 1, 2, 3, or 4.

For functions: 1 or 2.

For waveform memories: 1, 2, 3, or 4.

Example

The following program sets the FFT source to channel 1.

```
10  OUTPUT 707;":FFT:SOURCE CHANNEL1"
20  END
```

Query

```
FFT:SOURce?
```

The query returns the current source setting for the FFT.

Returned Format

```
FFT:SOURce {CHANnel<number> | FUNCtion<number | WMEMory<number
| <float_value>}<NL>
```

SPAN

Command

```
FFT:SPAN  <frequency_span_value>
```

This command sets the frequency span when the FFT is not in the magnify mode. This command will change the acquisition sample rate when in real time sampling mode. Sample rate is changed to be two times the frequency span value.

The magnify mode is controlled with the FFT:MAGNify command.

`<frequency_span_value>` Span number in Hertz.

Query

```
FFT:SPAN?
```

The query returns the current frequency span value for the FFT when not in magnify mode.

Returned Format

```
[FFT:SPAN] <frequency_span_value><NL>
```

Example

The following example places the current setting of the frequency span value in the string variable, Setting$, then prints the contents of the variable to the controller's screen.

```
10 DIM Setting$[50]
20 OUTPUT 707;":FFT:SPAN?"
30 ENTER 707;Setting$
40 PRINT Setting$
50 END
```

WINDow

Command

```
FFT:WINDow {RECTangular | HANNing | FLATtop}
```

This command sets the window type for the FFT. The FFT function assumes that the time record repeats. Unless there is an integral number of cycles in the sampled waveform in the record, a discontinuity is created at the end of the record. This introduces additional frequency components about the actual peaks. This is referred to as spectral leakage. In order to minimize spectral leakage, windows that approach zero smoothly at the beginning and end of the record are employed as filters to the FFTs.

Each window is useful for certain classes of input signals.

The RECTangular window is essentially no window, all points are multiplied by 1. The RECTangular window is useful for transient signals and signals where there are in integral number number of cycles in the time record. The HANNing window is useful for frequency resolution and general purpose use. It is good for resolving two frequencies that are close together or for making frequency measurements. The FLATtop window is best for making accurate amplitude measurements of frequency peaks.

Query

```
FFT:WINDow?
```

The query returns the current window for the FFT.

Returned Format

```
[FFT:WINDow] {RECTangular | HANNing | FLATtop}
```

Example

The following example places the current setting of the window in the string variable, Wind$, then prints the contents of the variable to the controller's screen.

```
10 DIM Wind$[50]
20 OUTPUT 707;":FFT:WINDOW?"
30 ENTER 707;Wind$
40 PRINT Wind$
50 END
```

27

Limit Test Commands

Limit Test Commands

The Limit Test commands and queries control the limit test features of the HP 54720 oscilloscope. Limit testing automatically compares measurement results with pass or fail limits. The limit test tracks up to four measurements. The action taken when the test fails is also controlled with commands in this subsystem.

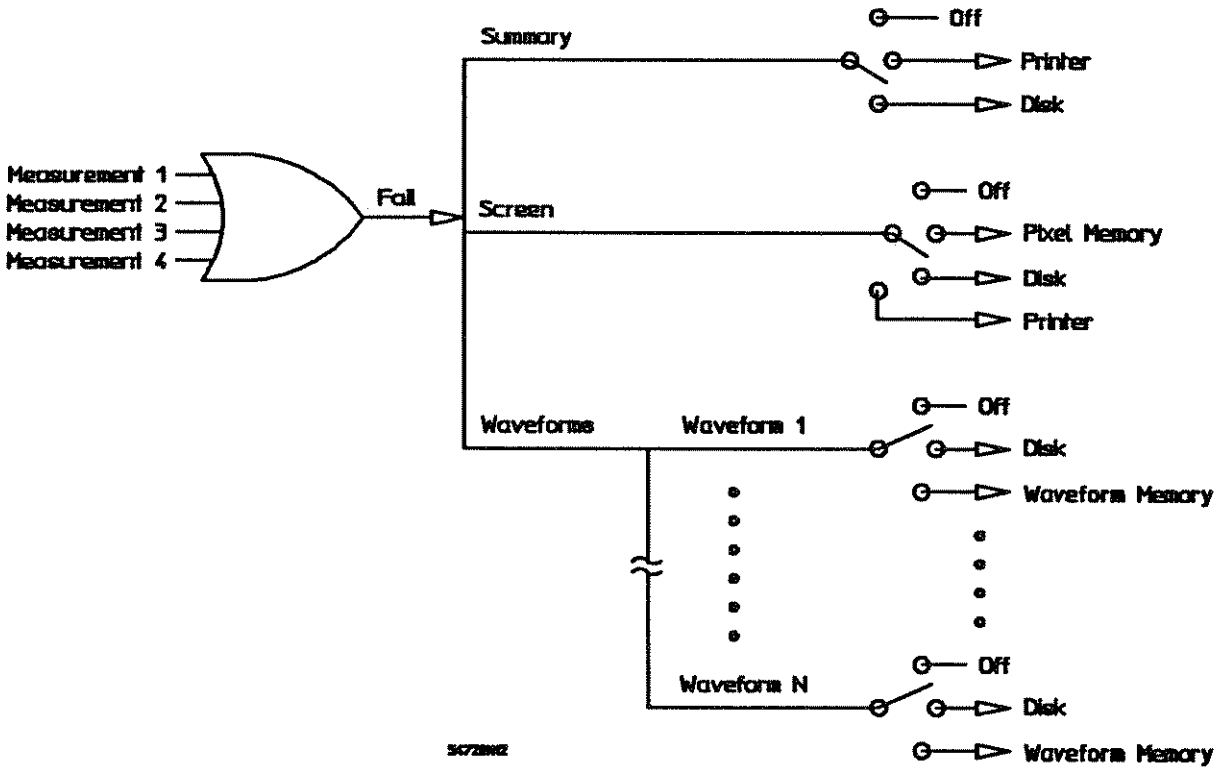
The Limit Test subsystem contains the following commands:

- FAIL,
- LLIMit (Lower Limit),
- MNFound (Measurement Not Found),
- RUN,
- SOURce,
- SSCReen (Store Screen),
- SSUMmary (Store Summary),
- SWAVeform (Store Waveform),
- TEST, and
- ULIMit (Upper Limit).

Figure 27-1 is a functional view of the limit test.

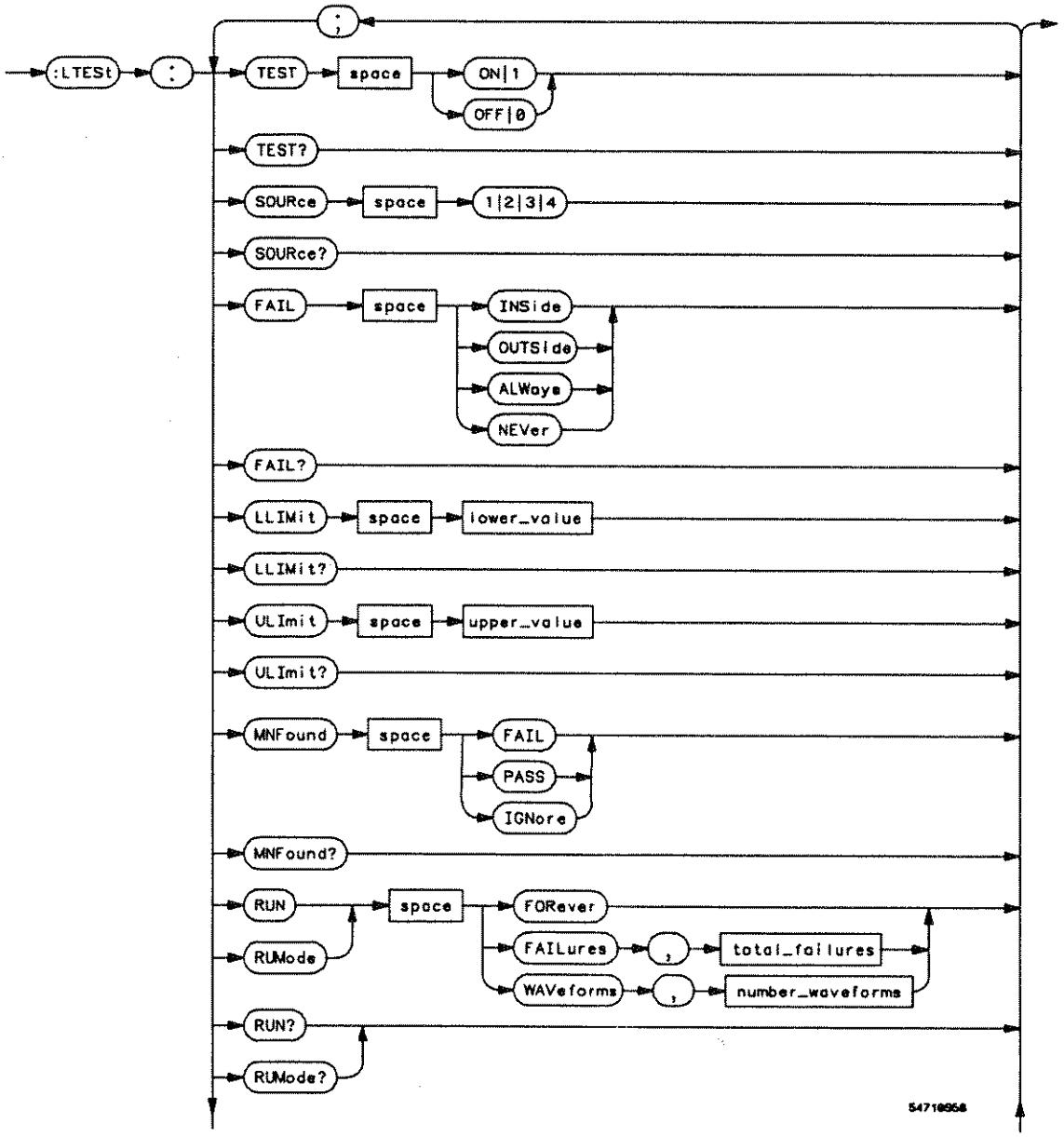
Figure 27-2 is the syntax diagram for the Limit Test subsystem commands.

Figure 27-1



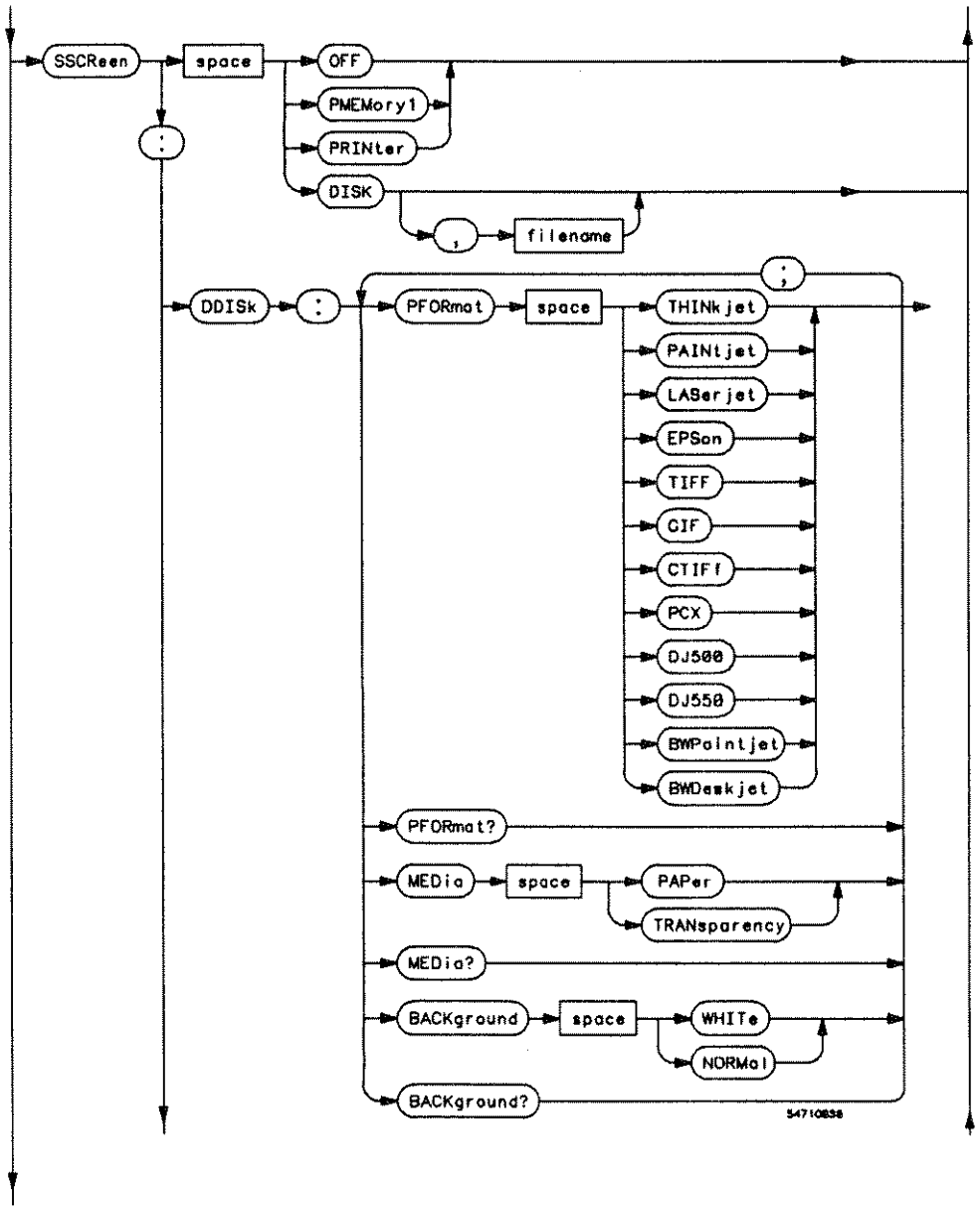
Limit Test Functional Diagram

Figure 27-2



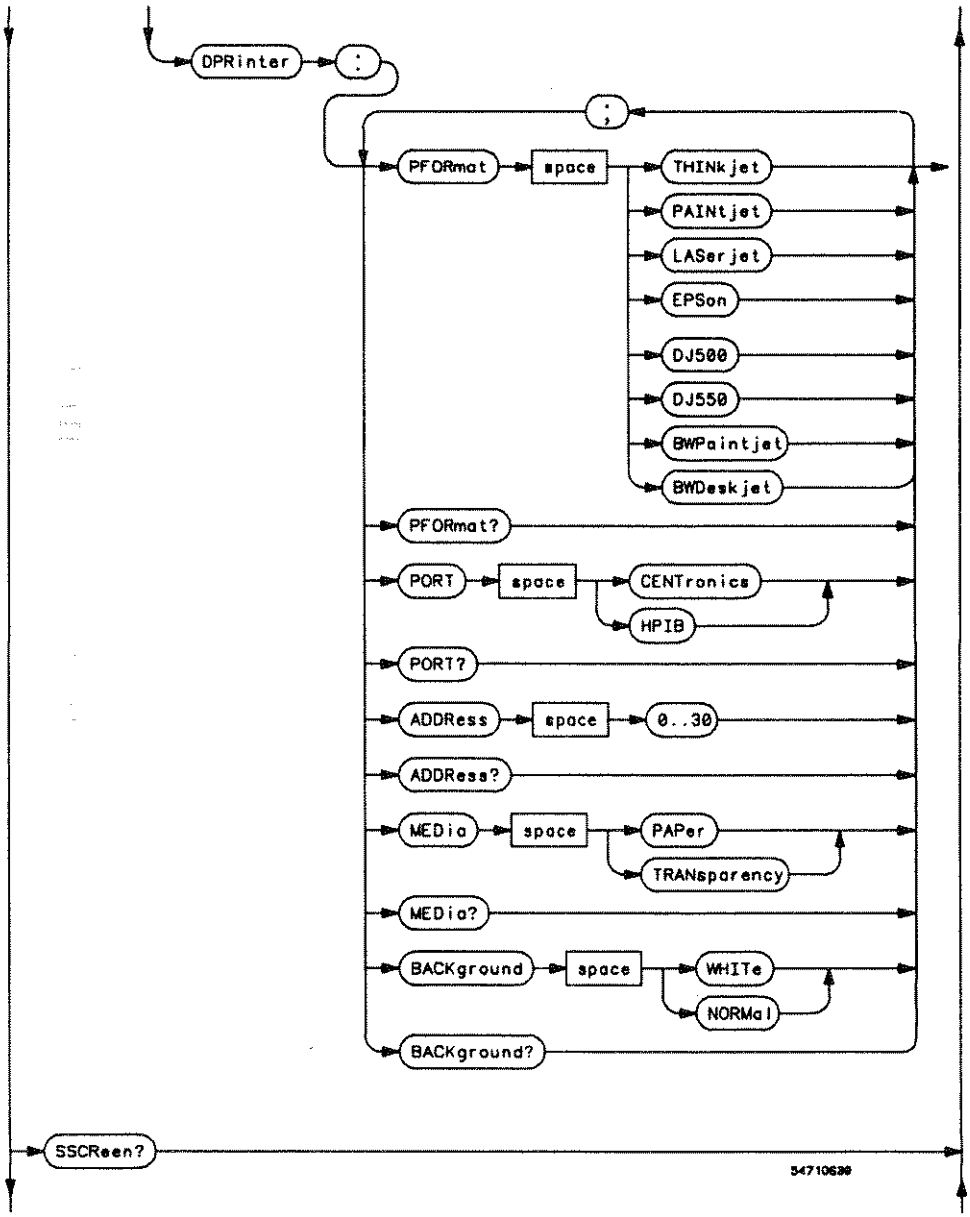
Limit Test Subsystem Syntax Diagram

Figure 27-2 (continued)



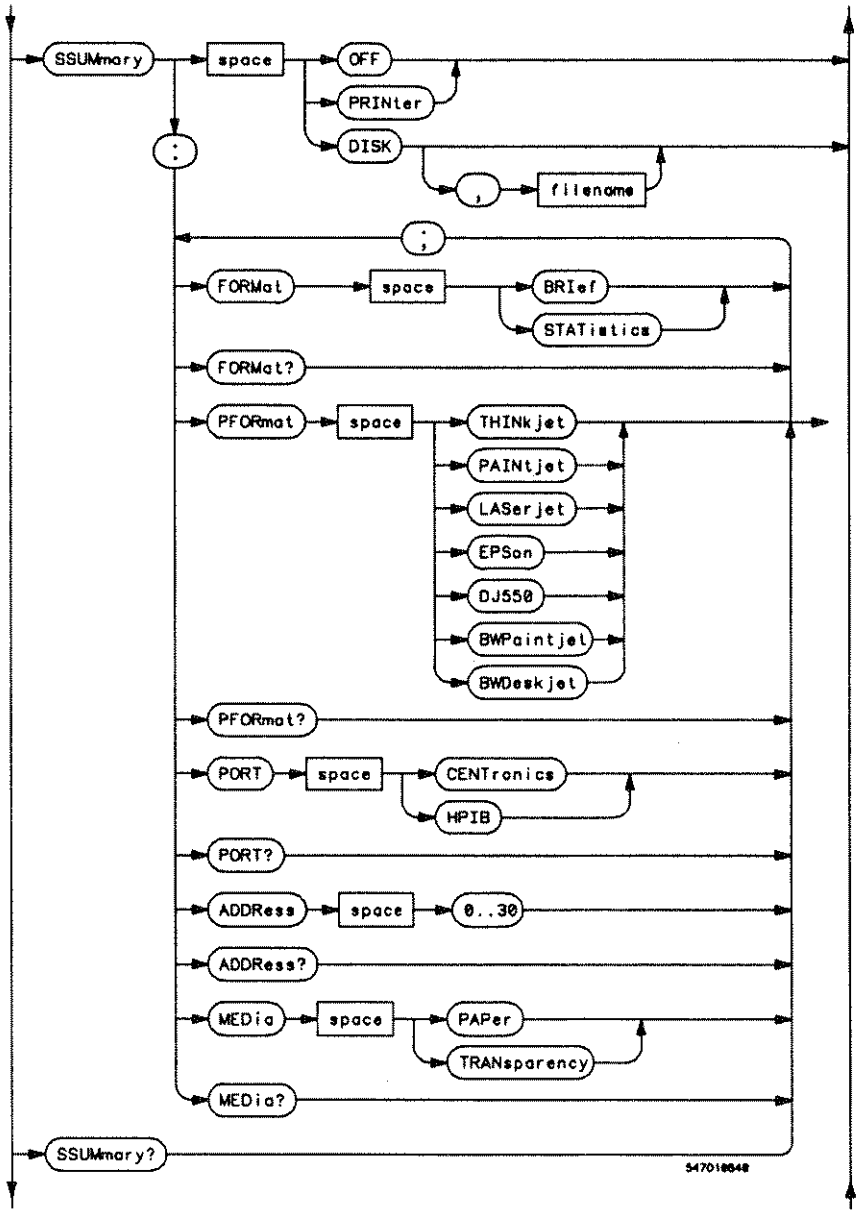
Limit Test Subsystem Syntax Diagram (continued)

Figure 27-2 (continued)



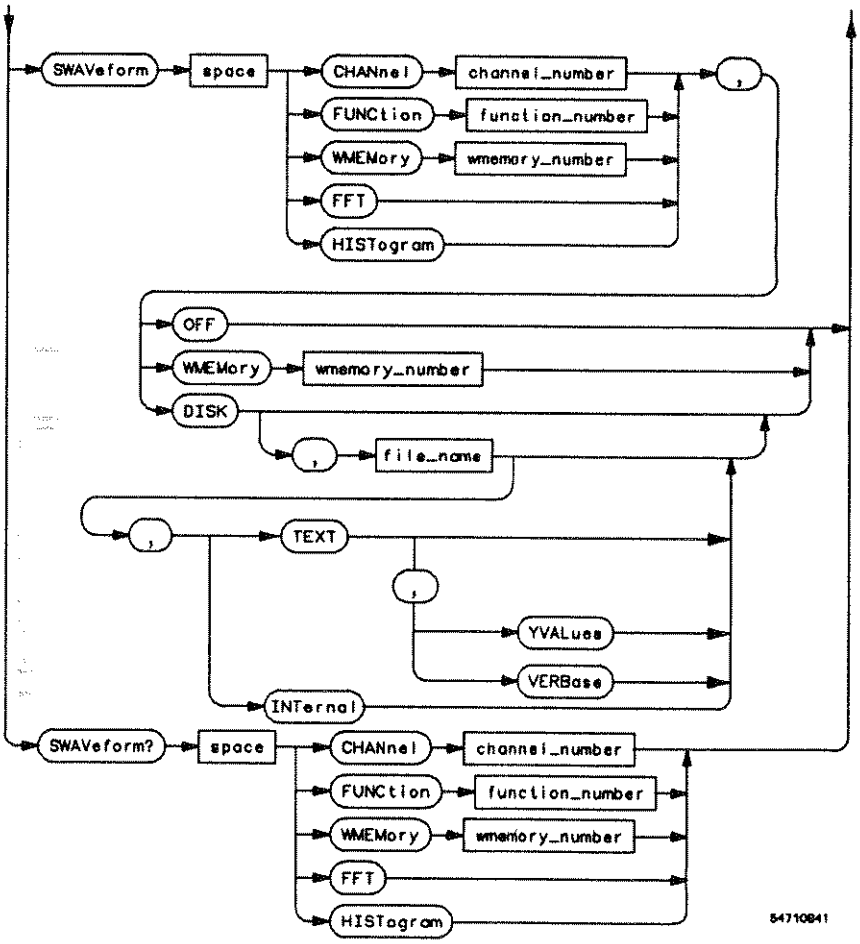
Limit Test Subsystem Syntax Diagram (continued)

Figure 27-2 (continued)



Limit Test Subsystem Syntax Diagram (continued)

Figure 27-2 (continued)



Limit Test Subsystem Syntax Diagram (continued)

FAIL

Command

```
:LTESt:FAIL {INSide | OUTSide | ALWays | NEVer}
```

The :LTESt:FAIL command sets the fail condition for an individual measurement. The conditions for a test failure are set on the source selected with the last LTESt:SOURce command.

When a measurement failure is detected by the limit test, the fail action conditions are executed, and there is the potential to generate an SRQ.

INSide FAIL:INside causes the oscilloscope to fail a test when the measurement results are within the parameters set by the LLTESt:LIMit and LTESt:ULIMit commands.

OUTSide FAIL:OUTside causes the oscilloscope to fail a test when the measurement results exceed the parameters set by LTESt:LLIMit and LTESt:ULIMit commands.

ALWays FAIL:ALWays causes the oscilloscope to fail a test every time the measurement is executed, and the parameters set by the LTESt:LLIMit and LTESt:uLIMit commands are ignored. The FAIL:ALWays mode logs the action each time the measurement is executed.

FAIL:ALWays can monitor trends in measurements. For example, tracking a measurement during an environmental test while the oscilloscope is running a measurement for a long time as the temperature or humidity is changed. Each time the measurement is executed, the results are logged as determined by the fail action set with the LTESt:SSCreen, LTESt:SSUMmary, or LTESt:SWAVeform commands.

NEVer FAIL:NEVer sets the oscilloscope so a measurement never fails a test. Use the FAIL:NEVer mode to observe one measurement but determine a failure from a different measurement.

The FAIL:NEVer mode monitors a measurement without any fail criteria.

Example

The following example causes the oscilloscope to fail a test when the measurements are outside the lower and upper limits.

```
10  OUTPUT 707;":LTEST:FAIL OUTSIDE"
20  END
```

Query

```
:LTESt:FAIL?
```

The query returns the current set fail condition.

Returned Format

```
[:LTESt:FAIL] {INSide | OUTSide | ALWays | NEVer}<NL>
```

Example

The following example returns the current fail condition and prints the result to the controller's screen.

```
10  DIM FAIL$[50]
20  OUTPUT 707;":LTEST:FAIL?"
30  ENTER 707;FAIL$
40  PRINT FAIL$
50  END
```

LLIMit

Command

```
:LTESt:LLIMit <lower_value>
```

The :LTESt:LLIMit (Lower LIMit) command sets the lower test limit for the active measurement currently selected by the :LTESt:SOURce command.

`<lower_value>` A real number.

Example

The following example sets the lower test limit to 1.

```
10  OUTPUT 707;":LTEST:LLIMIT 1"
20  END
```

If, for example, you chose to measure volts peak-peak and want the smallest acceptable signal swing to be one volt, you could use the above command, then set the limit test to fail when the signal is outside the specified limit.

Query

```
:LTESt:LLIMit?
```

The query returns the current value set by the command.

Returned Format

```
[:LTESt:LLIMit]<lower_value><NL>
```

Example

The following example returns the current lower test limit and prints the result to the controller's screen.

```
10  DIM LLIM$[50]
20  OUTPUT 707;":LTEST:LLIMIT?"
30  ENTER 707;LLIM$
40  PRINT LLIM$
50  END
```

MNFound

Command

```
:LTESt:MNFound {FAIL | PASS | IGNore}
```

The :LTESt:MNFound (Measurement Not Found) command sets the action to take when the measurement cannot be made. This command affects the active measurement currently selected by the last LTESt:SOURce command.

This command tells the instrument how to treat a measurement that cannot be made. For example, if a risetime between 1 to 5 volts is requested and the captured signal is between 2 to 3 volts, this control comes into play. Another use for this command is when trying to measure the frequency of a baseline waveform.

FAIL FAIL is used when the instrument cannot make a measurement. For example, when an edge is expected to be present and is not found. This is the mode to use for most applications.

The total number of waveforms is incremented, and the total number of failures is incremented.

PASS PASS might be used when triggering on one event and measuring another event which may not occur for every trigger. For example, in a communications test system, you might want to trigger on the clock and test the risetime of edges in the data stream. However, there may be no way to guarantee that a rising edge will be present to measure in the data stream at every clock edge. By using the PASS parameter, the limit test will not log a failure if there is no edge found in the data stream.

If the measurement cannot be made, the total number of waveforms measured is incremented, but the total number of failures is not.

IGNore IGNore is similar to PASS, except the totals for the number of waveforms and failures are not incremented. Therefore, the total indicates the number of tests when the measurement was made.

Example

The following example causes the oscilloscope to pass the test when a measurement cannot be made.

```
10  OUTPUT 707;":LTEST:MNFOUND PASS"
20  END
```

Query

```
:LTESt:MNFound?
```

The query returns the current action set by the command.

Returned Format

```
[:LTESt:MNFound] {FAIL | PASS | IGNore}<NL>
```

Example

The following example gets the current setting of the measurement not found action and prints the result to the controller's screen.

```
10  DIM MNF$[50]
20  OUTPUT 707;":LTEST:MNFOUND?"
30  ENTER 707;MNF$
40  PRINT MNF$
50  END
```

RUN (RUMode)

Command

```
:LTESt:RUN {FORever | FAILures, <total_failures> |
WAVeforms, <maximum_waveforms>}
```

The :LTESt:RUN command determines the termination conditions for the test. The choices are FORever, FAILures, or WAVeforms.

If FAILures or WAVeforms are selected a second parameter is required indicating the number of failures that can occur or the number of waveforms that are to be acquired.

The keyword RUMode (Run Until Mode) may be used instead of RUN.

FORever FORever runs the limit test until the test is turned off. This mode is used when you want a measurement to run continually and not to stop after a fixed number of failures. For example, you may want the limit test to run overnight and not be limited by a number of failures.

FAILures FAILures runs the limit test until a set number of failures occurs. When FAILures is sent, the test executes until the selected total failures are obtained. The number of failures are compared against this number to test for termination.

Use the FAILures mode when you want the limit test to reach completion after a set number of failures. The total number of failures is additive for all of the measurements. For example, if you select 10 failures, the total of 10 failures can come from several measurements. The 10 failures can be the sum of four rise time failures, four + width failures, and two Overshoot failures.

<total_failures> An integer: 1 to 1,000,000,000.

WAVeforms WAVeforms sets the maximum waveforms that are required. When any measurement reaches this number of waveforms the test terminates. WAVeforms runs the limit test until a set number of waveforms are acquired. In the real-time acquisition mode, a waveform is acquired with each trigger event. That is because in the real-time acquisition mode, all the data points that make up a waveform are acquired from a single trigger event. If you select 10 waveforms, that is the same as 10 trigger events.

In the equivalent-time sampling mode, a waveform is reconstructed from several trigger events. The limit test operates in conjunction with the

ACQuire:COMPlete command. If the ACQuire:COMPlete is set to OFF, measurements and limit test consider each trigger event as a waveform.

Data from previous triggers is still retained in memory and the latest trigger may or may not add additional data points to the waveform.

When ACQuire:COMPlete is ON, the command acts like a holdoff control because the measurement is held off until after the completion criteria is met. If limit test is on, it also waits for a measurement before testing the measurement results. For example, if completion is set to 95%, then measurements are performed on the data after 95% of the waveform record is filled with data. Depending on the setup of the oscilloscope, it can take several triggers before the completion criteria is met. After the completion criteria is met, measurements are performed and limit test checks the measurement results. The data is then cleared from memory, and the oscilloscope starts acquiring new data.

> Complete percentage is set with the ACQuire:COMPlete command.

Use the waveforms mode when you want the limit test to reach completion after a set number of waveforms are acquired. The test terminates with the measurement that first reaches the specified number of waveforms.

`<maximum_waveforms>` An integer: 1 to 1,000,000,000.

Example

The following example causes limit test to run until 25 waveforms are acquired.

```
10  OUTPUT 707;":LTEST:RUN WAVEFORMS,25"
20  END
```

Query

```
:LTESt:RUN?
```

The query returns the currently selected termination condition and value.

Returned Format

```
[:LTESt:RUN] {FOREver | FAILures, <total_failures> |
WAVeforms, <maximum_waveforms>}<NL>
```

Example

The following example returns the current condition under which the limit test terminates and prints the result to the controller's screen.

```
10  DIM RUN$[50]
20  OUTPUT 707;":LTEST:RUN?"
30  ENTER 707;RUN$
40  PRINT RUN$
50  END
```

SOURce

Command

```
:LTESt:SOURce  {1 | 2 | 3 | 4}
```

The :LTESt:SOURce command selects the current source for the ULIMit, LLIMit, MNFound, and FAIL commands. It selects one of the active measurements as referred to by their position in the measurement window on the bottom of the screen. Source 1 is the measurement on the top line, 2 is on the second line, and so on.

Example

The following example selects the first measurement as the source for the limit testing commands.

```
10  OUTPUT 707;":LTEST:SOURCE 1"
20  END
```

Query

```
:LTESt:SOURce?
```

The query returns the currently selected measurement source.

Returned Format

```
[:LTESt:SOURce]  {1 | 2 | 3 | 4} <NL>
```

Example

The following example returns the currently selected measurement source for the limit testing commands.

```
10  DIM SOURCE$[50]
20  OUTPUT 707;":LTEST:SOURCE?"
30  ENTER 707;SOURCE$
40  PRINT SOURCE$
50  END
```

See Also

Measurements are started in the Measurement Subsystem.

SSCReen

Command

:LTESt:SSCReen {OFF | PMEMory1 | PRINter | DISK [,<filename>]}

The :LTESt:SSCReen (Store SCReen) command saves a copy of the screen in the event of a failure.

A destination of OFF turns off the save action.

A destination of PMEMory1 adds the screen to the pixel memory.

When the destination is PRINter, a variety of printer-related controls are used to specify the printer. When the destination is DISK, a different set of commands is provided to control the print to disk. The two printer specifications are set through the DPRinter (Destination Printer) and the DDISk (Destination Disk) commands.

<filename> A file prefix of four characters. If no prefix is specified, the default prefix is SCRN.

The LTESt:SSCReen contains the following command sub-subsystems when either PRINter or DISK is specified:

- DDISk (Display Disk), and
- DPRinter (Display Printer).

Example

The following example saves a copy of the screen to the printer in the event of a failure. Additional printer-related controls are set using the SSCReen:DPRinter command set.

```
10  OUTPUT 707;":LTEST:SSCREEN PRINTER"
20  END
```

Query

```
:LTESt:SSCReen?
```

The query returns the current state of the SSCreen command.

Returned Format

```
[:LTESt:SSCReen]{OFF | PMEMory1 | PRINter | DISK
[,<filename>]} <NL>
```

Example

The following example returns the destination of the save screen when a failure occurs and prints the result to the controller's screen.

```
10  DIM SSCR$[50]
20  OUTPUT 707;":LTEST:SSCREEN?"
30  ENTER 707;SSCR$
40  PRINT SSCR$
50  END
```

SSCReen:DDISk

The LTESt:SSCReen:DDISk sub-subsystem commands are used to set up the disk drive when storing the limit test display screen to a disk. Use DDISk to store limit test display screens to a file in printer format for later printing. The disk drive setup consists of the following commands:

- BACKground,
- MEDia, and
- PFORmat (printer format).

SSCReen:DDISk:BACKground

Command

```
:LTESt:SSCReen:DDISk:BACKground {WHITe | NORMal}
```

This command controls the background color of the graticule area of an HP PaintJet print. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the disk.

Example

The following example sets the background color of the HP PaintJet print to white.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:BACKGROUND WHITE"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:BACKground?
```

The query returns the current background for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:BACKground] {WHITe | NORMal}<NL>
```

Example

The following example returns the current background color of the HP PaintJet print and prints the result to the controller's screen.

```
10  DIM DDBCK$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:BACKGROUND?"
30  ENTER 707;DDBCK$
40  PRINT DDBCK$
50  END
```

SSCReen:DDISk:MEDia

Command

```
:LTESt:SSCReen:DDISk:MEDia {PAPer | TRANsparency }
```

The MEDia command specifies either paper or transparency to be used in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This control is used when the store screen is directed to the disk, and is valid only when printing to the HP PaintJet and Color DeskJet printers.

Example

The following example specifies that paper will be used in the printer when a file saved to disk is later printed.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:MEDIA PAPER"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:MEDia?
```

The query returns the current media for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example returns the current media to be used in the printer and prints the result to the controller's screen.

```
10  DIM DDMED$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:MEDIA?"
30  ENTER 707;DDMED$
40  PRINT DDMED$
50  END
```

SSCReen:DDISk:PFORmat

Command

```
:LTESt:SSCReen:DDISk:PFORmat { THINkjet | PAINtjet
| LASerjet | EPSon | GIF | TIFF | CTIFf | PCX |
DJ500 | DJ550 | BWPaintjet | BWDeskjet}
```

The PFORmat (Print FORmat) command selects the file format for a stored file or the printer format to use when storing the screen to a disk for later printing. This includes all print formats supported by the instrument.

Example

The following example sets the format of the file stored to disk for later printing to an HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSCREEN:DDISK:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSCReen:DDISk:PFORmat?
```

The query returns the current printer format for the disk store.

Returned Format

```
[:LTESt:SSCReen:DDISk:PFORmat] { THINkjet | PAINtjet |
LASerjet | EPSon | GIF | TIFF | CTIFf | PCX | DJ500 | DJ550 |
BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the file format and prints the result to the controller's screen.

```
10  DIM DDPFORM$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DDISK:PFORMAT?"
30  ENTER 707;DDPFORM$
40  PRINT DDPFORM$
50  END
```

SSCReen:DPRinter

The LTESt:SSCReen:DPRinter sub-subsystem commands are used to set up the printer when storing the limit test display screen to a printer. The printer setup consists of the following commands:

- ADDRess,
- BACKground,
- MEDia,
- PFORmat, and
- PORT.

SSCReen:DPRinter:ADDRess

Command

```
:LTESt:SSCReen:DPRinter:ADDRess <address_value>
```

The ADDRess command allows the user to select the HP-IB address for the printer. This address is used only if the port is HP-IB.

`<address_value>` Any HP-IB address 0-30.

Example

The following example sets the HP-IB address for the printer to 1.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:ADDRESS 1"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:ADDRess?
```

The query returns the current HP-IB address of the printer.

Returned Format

```
[:LTESt:SSCReen:DPRint:ADDRess]<address_value><NL>
```

Example

The following example returns the current HP-IB address for the printer and prints the result to the controller's screen.

```
10  DIM DPADD$[10]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:ADDRESS?"
30  ENTER 707;DPADD$
40  PRINT DPADD$
50  END
```

SSCReen:DPRinter:BACKground

Command

```
:LTESt:SSCReen:DPRinter:BACKground {WHITe | NORMal}
```

This command controls the background color of the graticule area of an HP PaintJet print. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the printer.

Example

The following example sets the background to white.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:BACKGROUND WHITE"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:BACKground?
```

The query returns the current background for the printer.

Returned Format

```
[:LTESt:SSCReen:DPRinter:BACKground] {WHITe | NORMal} <NL>
```

Example

The following example returns the current background setting and prints the result on the controller screen:

```
10  DIM FAIL$[10]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:BACKGROUND?"
30  ENTER 707;FAIL$
40  PRINT FAIL$
50  END
```

SSCReen:DPRinter:MEDia

Command

```
:LTESt:SSCReen:DPRinter:MEDia {PAPer | TRANsparency}
```

The MEDia command specifies either paper or transparency in the printer. This control is used when the store screen is directed to a printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This command is valid only for the HP PaintJet and Color DeskJet printers.

Example

The following example selects paper as the printer media.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:MEDIA PAPER"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:MEDia?
```

The query returns the current printer media.

Returned Format

```
[:LTESt:SSCReen:DPRinter:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following example gets the current media setting and prints the result to the controller's screen.

```
10  DIM DPMED$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:MEDIA?"
30  ENTER 707;DPMED$
40  PRINT DPMED$
50  END
```

SSCReen:DPRinter:PFORmat

Command

```
:LTESt:SSCReen:DPRinter:PFORmat {THINkjet |
PAINtjet | LASerjet | EPSon | DJ500 | DJ550 |
BWPaintjet | BWDeskjet}
```

The PFORmat (Printer FORmat) command selects the printer format to use when storing the screen to a printer. This includes only the formats available from the setup print menu that refer to a physical printer.

Example

The following example sets the printer format to an HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:PFORmat?
```

The query returns the current printer format.

Returned Format

```
[:LTESt:SSCReen:DPRinter:PFORmat] {THINkjet | PAINtjet |
LASerjet | EPSon | DJ500 | DJ550 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the current printer format and prints the result to the controller's screen.

```
10  DIM DPPFORM$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PFORMAT?"
30  ENTER 707;DPPFORM$
40  PRINT DPPFORM$
50  END
```

SSCReen:DPRinter:PORT

Command

```
:LTESt:SSCReen:DPRinter:PORT {CENTronics | HPIB}
```

The PORT command selects the printer port for the screen when store screen is going to the printer.

Example

The following example selects the Centronics printer port.

```
10  OUTPUT  707;":LTEST:SSCREEN:DPRINTER:PORT CENT"
20  END
```

Query

```
:LTESt:SSCReen:DPRinter:PORT?
```

The query returns the current port type.

Returned Format

```
[:LTESt:SSCReen:DPRinter:PORT]{CENTronics | HPIB} <NL>
```

Example

The following example returns the current printer port and prints the result to the controller's screen.

```
10  DIM DPPORT$[50]
20  OUTPUT 707;":LTEST:SSCREEN:DPRINTER:PORT?"
30  ENTER 707;DPPORT$
40  PRINT DPPORT$
50  END
```

SSUMmary

Command

```
:LTESt:SSUMmary {OFF | PRINter | DISK
[,<filename>]}
```

The :LTESt:SSUMmary (Store SUMmary) command saves the summary in the event of a failure.

A destination of OFF turns off the summary save.

A destination of PRINter builds the summary file and sends it to the printer. In this case a variety of printer-related controls are used to specify the printer configuration. The following printer commands are used when storing the summary to a printer:

- ADDRess,
- FORMat,
- MEDia,
- PFORmat (printer format), and
- PORT.

When set to DISK, the summary is written to the disk drive using a user-specified base name with a machine generated suffix. For example, the results of tests 1, 2, and 3 may be written to files SUMM0001.SUM, SUMM0002.SUM, and SUMM0003.SUM. The summary is a logging method where the user can get an overall view of the test results. The summary is an ASCII file that the user can peruse on a computer, read into a spreadsheet, etc.

`<filename>` A file prefix of four characters defaulted to SUMM.

Example

The following example saves the summary to a disk file named TEST0000.SUM.

```
10  OUTPUT 707;":LTEST:SSUMMARY DISK,TEST"
20  END
```

Query

```
:LTESt:SSUMmary?
```

The query returns the current specified destination for the summary.

Returned Format

```
[:LTESt:SSUMmary] {OFF | PRINter | DISK [,<filename>]}<NL>
```

Example

The following example returns the current destination for the summary and and prints the result to the controller's screen.

```
10  DIM SUMM$[50]
20  OUTPUT 707;":LTEST:SUMMARY?"
30  ENTER 707;SUMM$
40  PRINT SUMM$
50  END
```

SSUMmary:ADDRess

Command

```
:LTESt:SSUMmary:ADDRess <address_value>
```

The :LTESt:SSUMmary:ADDRess command selects the HP-IB address for the printer. This address is used only when the summary is going to the printer and the Port is set to HP-IB.

`<address_value>` Any HP-IB address from 0 to 30.

Example

The following example sets the HP-IB address for the printer to 3.

```
10  OUTPUT 707;":LTEST:SSUMMARY:ADDRESS 1"
20  END
```

Query

```
:LTESt:SSUMmary:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:LTESt:SSUMmary:ADDRess]<address_value><NL>
```

Example

The following example returns the current HP-IB address of the printer and prints the result to the controller's screen.

```
10  DIM SUMMADD$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:ADDRESS?"
30  ENTER 707;SUMMADD$
40  PRINT SUMMADD$
50  END
```

SSUMmary:FORMat

Command

```
:LTESt:SSUMmary:FORMat {BRIef | STAtistics}
```

This command specifies whether the summary file is either brief or contains statistics.

BRIef is a one line log of the failure.

STAtistics includes the measurement statistics plus the one line log.

Example

The following example specifies a brief summary file.

```
10  OUTPUT 707;":LTEST:SSUMMARY:FORMAT BRIEF"
20  END
```

Query

```
:LTESt:SSUMmary:FORMat?
```

Returned Format

```
[:LTESt:SSUMmary:FORMat] {BRIef | STATistics}<NL>
```

Example

The following example returns the current summary file format and prints the result to the controller's screen.

```
10  DIM SUMMFORM$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:FORMAT?"
30  ENTER 707;SUMMFORM$
40  PRINT SUMMFORM$
50  END
```

SSUMmary:MEDia

Command

```
:LTESt:SSUMmary:MEDia {PAPer | TRANsparency}
```

The :LTESt:SSUMmary:MEDia command specifies whether paper or transparency film will be used in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the media look darker and slows down the printing process. This control is used when the store summary is directed to the printer, and is valid only for the HP PaintJet and Color DeskJet printers.

Example

The following example specifies that transparency media should be in the printer when the summary is printed.

```
10  OUTPUT 707;":LTEST:SSUMMARY:MEDIA TRANSPARENCY"
20  END
```

Query

```
:LTESt:SSUMmary:MEDia?
```

The query returns the current specified media for the printer.

Returned Format

```
[:LTESt:SSUMmary:MEDia]{PAPer | TRANsparency}<NL>
```

Example

The following example returns the current media requirement for the printer and prints the result to the controller's screen.

```
10  DIM SUMMMED$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:MEDIA?"
30  ENTER 707;SUMMMED$
40  PRINT SUMMMED$
50  END
```

SSUMmary:PFORmat

Command

```
:LTESt:SSUMmary:PFORmat {THINkjet | PAINtjet |
LASerjet | EPSon | DJ550 | BWPaintjet | BWDeskjet}
```

The :LTESt:SSUMmary:PFORmat (Print FORmat) command selects the printer format to use when storing the summary to a printer. This includes the physical printers supported by the instrument with the exception of the HP DeskJet 500C with the color cartridge installed. To set up the format for the HP DeskJet 500C, use the monochrome cartridge and the BWDeskjet parameter in the command.

Example

The following example sets the printer format for the summary file to HP PaintJet.

```
10  OUTPUT 707;":LTEST:SSUMMARY:PFORMAT PAINTJET"
20  END
```

Query

```
:LTESt:SSUMmary:PFORmat?
```

The query returns the current specified printer format.

Returned Format

```
[:LTESt:SSUMmary:PFORmat] {THINkjet | PAINtjet | LASerjet |
EPSon | DJ550 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example returns the current printer format for the summary file and prints the result to the controller's screen.

```
10  DIM SUMMPFORM$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:PFORMAT?"
30  ENTER 707;SUMMPFORM$
40  PRINT SUMMPFORM$
50  END
```

SSUMmary:PORT

Command

```
:LTESt:SSUMmary:PORT {CENTronics | HPIB}
```

The :LTESt:SSUMmary:PORT command selects the printer port for the summary when the summary is sent to the printer.

Example

The following example sets the printer port for the summary to HP-IB.

```
10  OUTPUT 707;":LTEST:SSUMMARY:PORT HPIB"
20  END
```

Query

```
:LTESt:SSUMmary:PORT?
```

The query returns the current specified port for the printer.

Returned Format

```
[:LTESt:SSUMmary:PORT]{CENTronics | HPIB} <NL>
```

Example

The following example returns the current printer port for the summary file and prints the result to the controller's screen.

```
10  DIM SUMMPORT$[50]
20  OUTPUT 707;":LTEST:SSUMMARY:PORT?"
30  ENTER 707;SUMMPORT$
40  PRINT SUMMPORT$
50  END
```

SWAVeform

Command

```
:LTESt:SWAVeform  <source>, <destination>,
[<filename>[,<format>]]
```

The :LTESt:SWAVeform (Store WAVeform) command saves waveforms from a channel, function, histogram, or waveform memory in the event of a failure detected by the limit test. Each waveform source can be individually specified allowing multiple channels or functions to be saved to disk or waveform memories. Setting a particular source to OFF removes any waveform save action from that source.

<source> One of {CHANnelN | FUNCtionN | FFT | HISTogram | WMEMoryN}

<destination> One of {OFF | WMEMoryN | DISK}

<filename> A descriptive file prefix consisting of up to four characters. If no filename is specified, the prefix defaults to CH1A...CH4B, FN1, FN2, FFT, HIST, MEM1..MEM4.

<format> One of {TEXT [,YVALues> | VERBose] | INTernal}

Example

The following example turns off the saving of waveforms from channel 1 in the event of a limit test failure.

```
10  OUTPUT 707;":LTEST:SWAVEFORM CHAN1,OFF"
20  END
```

Query

```
:LTESt:SWAVeform? <source>
```

The query returns the current state of the :LTESt:SWAVeform command.

Returned Format

```
[:LTESt:SWAVeform]<source>, <destination>,
[<filename>[,<format>]]<NL>
```

Example

The following example returns the current parameters for saving waveforms in the event of a limit test failure.

```
10  DIM SWAV$[50]
20  OUTPUT 707;":LTEST:SWAVEFORM? CHANNEL1"
30  ENTER 707;SWAV$
40  PRINT SWAV$
50  END
```

TEST

Command

```
:LTESt:TEST {{ON | 1} {OFF | 0}}
```

The LTESt:TEST command controls the execution of the limit test function. ON allows the limit test to run over all of the active measurements. When the limit test is turned on, the limit test results are displayed on screen in a window below the graticule.

Example

The following example turns off the limit test function.

```
10  OUTPUT 707;":LTEST:TEST OFF"
20  END
```

Query

```
:LTESt:TEST?
```

The query returns the state of the TEST control.

Returned Format

```
[:LTESt:TEST] {1 | 0} <NL>
```

Example

The following example returns the current state of the limit test (on or off, 1 or 0, respectively) and prints the result to the controller's screen.

```
10  DIM TEST$[50]
20  OUTPUT 707;":LTEST:TEST?"
30  ENTER 707;TEST$
40  PRINT TEST$
50  END
```

The results of the MEAS:RESults? query have three extra fields when LimitTESt:TEST is ON (failures, total, status). Failures is a number, total is a number, and status is one of the following values:

0 OK,
1 failed high,
2 failed low,
3 failed inside , and
4 other failures.

ULIMit

Command

```
:LTESt:ULIMit <upper_value>
```

The :LTESt:ULIMit (Upper LIMit) command sets the upper test limit for the active measurement currently selected by the last :LTESt:SOURce command.

`<upper_value>` A real number.

Example

The following example sets the upper limit of the currently selected measurement to 500 milli.

```
10  OUTPUT 707;":LTEST:ULIMIT 500E-3"
20  END
```

Suppose you are measuring the maximum voltage of a signal with Vmax, and that voltage should not exceed 500 mV. You can use the above program and set the LTESt:FAIL OUTside command to specify that the limit subsystem will fail a measurement when the voltage exceeds 500 mV.

Query

```
:LTESt:ULIMit?
```

The query returns the current upper limit of the limit test.

Returned Format

```
[:LTESt:ULIMit] <upper_value><NL>
```

Example

The following example returns the current upper limit of the limit test and prints the result to the controller's screen.

```
10  DIM ULIM$[50]
20  OUTPUT 707;":LTEST:ULIMIT?"
30  ENTER 707;ULIM$
40  PRINT ULIM$
50  END
```

28

Mask Test Commands

Mask Test Commands

The Mask Test commands and queries control the mask test features of the HP 54720 oscilloscope. Mask Testing automatically compares measurement results with the boundaries of a set of polygons that you define. Any waveform or sample that falls within the boundaries of one or more polygons is recorded as a failure.

The Mask Test subsystem contains the following commands:

- AMASk,
- COUNt,
- MASK,
- POLYgon,
- RUMode,
- SCALe,
- SSCReen,
- SSUMmary,
- SWAVeform, and
- TEST.

Building Polygon Masks

The oscilloscope allows you to build a mask using either polygons or a reference waveform. With the polygon method, you use polygons to mask off failure regions on the graticule area. You can position up to eight polygons on the graticule area, and each polygon can have from 3 to 512 sides. You use the :MTESt:POLYgon:DEFine and :MTESt:MASk:DEFine commands to define polygons.

The polygon method is typically used for telecommunications applications because of the flexibility polygons give in designing a mask. For example, you can construct very complicated masks with polygons, or you can place polygons within polygons. Placing polygons within polygons allows you to test waveform failure rates to

different tolerances because you can obtain separate failure statistics for each polygon, using the :MTESt:COUNt:FAILures? query. For example, an outer polygon could represent a 1% tolerance, and two inner polygons could represent a 5% tolerance and 10% tolerance respectively.

With the reference waveform method, you construct a mask by adding a ΔX and ΔY tolerance around your reference waveform, using the :MTESt:AMASk subsystem commands. The reference waveform method is simpler to use but less flexible than the polygon method.

Mask Handling in the Oscilloscope

The oscilloscope has three features that use a specific database. This database uses a different memory area than the waveform record for each channel. The three features that use the database are histograms, mask testing, and color-graded display. When any one of these three features is turned on, the oscilloscope starts building the database. The database is the size of the graticule area, which is 256 pixels high by 451 pixels wide. Behind each pixel is a 16-bit counter. Each counter is incremented each time a pixel is hit by data from a channel or function. The maximum count (saturation) for each counter is 65,535. You can check to see if any of the counters is close to saturation by using the DISPlay:CGRade ON command to enable the color-graded display feature. The color-graded display uses colors to represent the number of hits on various areas of the display.

The database continues to build until the oscilloscope stops acquiring data or all three functions (color-graded display, mask testing, and histograms) are turned off. The oscilloscope stops acquiring data when the power is cycled, the Stop/Single hardkey is pressed, or the run until softkey in the mask or histograms menu is set to stop acquiring data after a specified number of waveforms or samples are acquired.

You can clear the database by pressing the Clear display hardkey, cycling the power, turning off all three features that use the database, or sending a CDISplay command. The database does not differentiate waveforms from different channels or functions. If three channels are turned on and the waveform for each channel happens to light the

same pixel at the same time, the counter is incremented by three. However, you cannot tell how many hits came from each waveform. You can separate waveforms by setting the display to two graphs and by positioning the waveforms vertically with the channel offset. By separating the waveforms you can avoid overlapping data in the database caused by multiple waveforms.

Mask File Format

Because polygon masks can be complicated, it's usually easier to define them and save them in a disk file, then retrieve them from the disk when needed for a measurement. You load a mask file using the DISK:LOAD command. Mask files have the extension .msk and have four parts:

- A mask title up to twenty characters (optional). This title will be displayed below the graticule when the mask is loaded.
- A polygon identifier, 1 through 8.
- The number of vertices for the polygon being defined.
- A series of X-Y coordinates defining the vertices. These are floating point numbers. The special values MIN and MAX automatically set a vertex X or Y coordinate to the boundary of the graticule, even if the scaling is changed with the :MTESt:SCALe commands.

Mask files can contain "C" style comments, which are enclosed in /* and */. Commas, tabs, spaces, or newlines are valid separators between parameters.

The following is a sample mask file for the DS1 E telecommunications waveform:

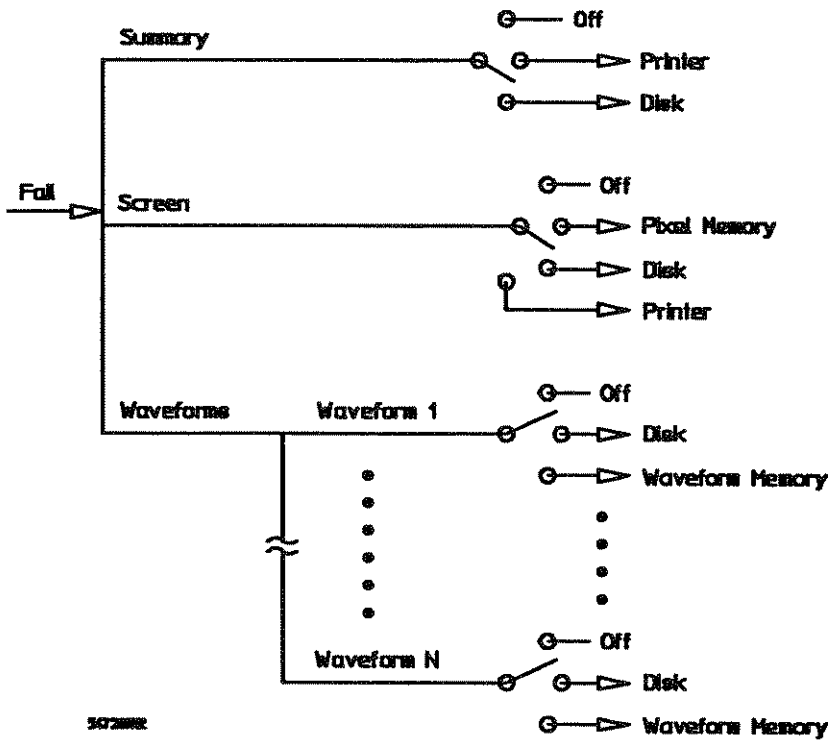
```
"DS1Eur 2048 kbit/s"
/*
     Physical/Electrical Characteristics of Heirarchical
     Digital Interface. (Geneva 1972 : further ammended)
     Recommendation G.703
*/
/*   Level1 Peak 2.37V   */
/*   Level2      0.00V   */
/*   Delta X     244 ns  */

/* Top Polygon           */   1
/* Number of vertices    */   9
     -0.5,      Max      /* Top of screen left side     */
     -0.5,     +0.1
     -0.0512,  +0.5
     -0.0512,  +1.2
     +0.5,     +1.1
     +1.0512,  +1.2
     +1.0512,  +0.5
     +1.5,     +0.1
     +1.5,      Max      /* Top of screen right side    */
/*_____________________________________________________*/

/* Bottom Polygon        */   2
/* Number of vertices    */   11
     -0.5,      Min      /* Bottom of screen left side  */
     -0.5,     -0.1
     +0.05123, -0.2
     +0.05123, +0.5
     +0.10246, +0.8
     +0.5,     +0.9
     +0.8975,  +0.8
     +0.94877, +0.5
     +0.94877, -0.2
     +1.5,     -0.1
     +1.5,      Min     /* Bottom of screen right side */
/*_____________________________________________________*/
```

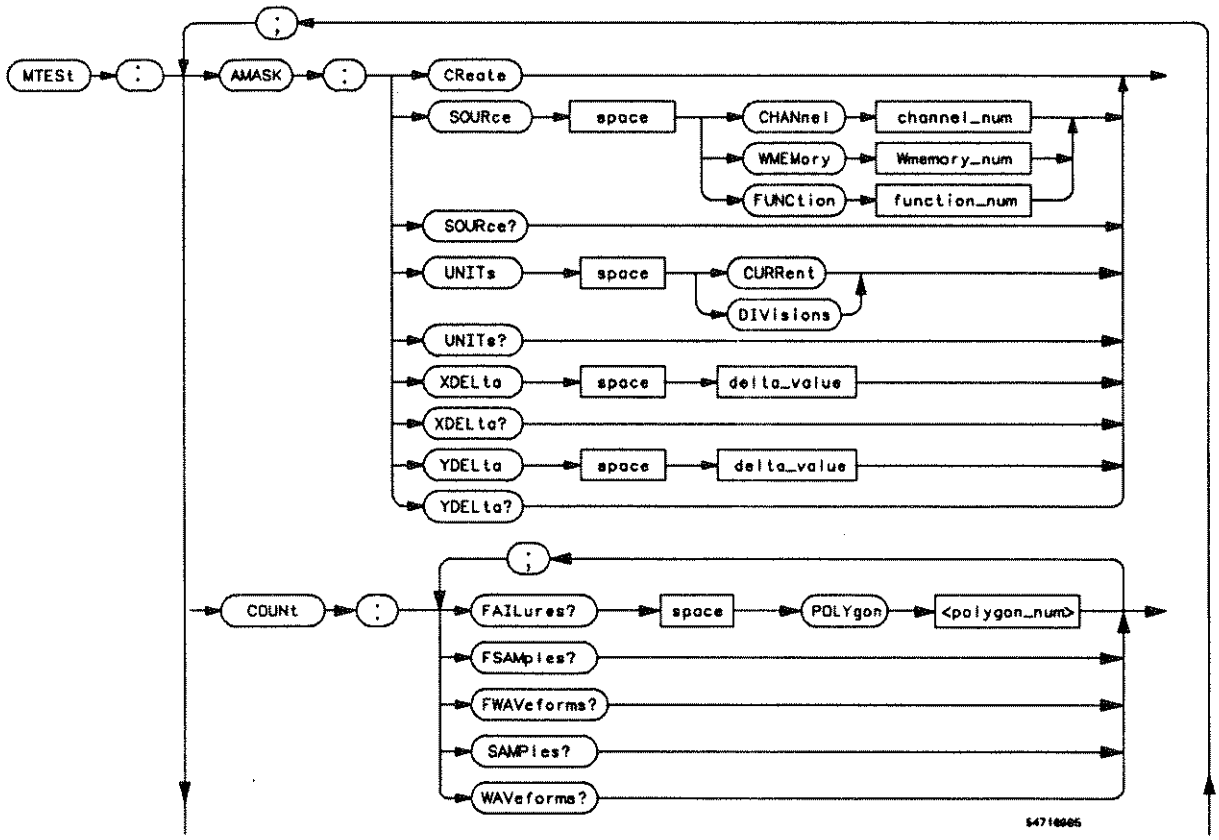
Figure 28-1 is a functional view of the Mask Test. Figure 28-2 is the syntax diagram for the Mask Test subsystem commands.

Figure 28-1



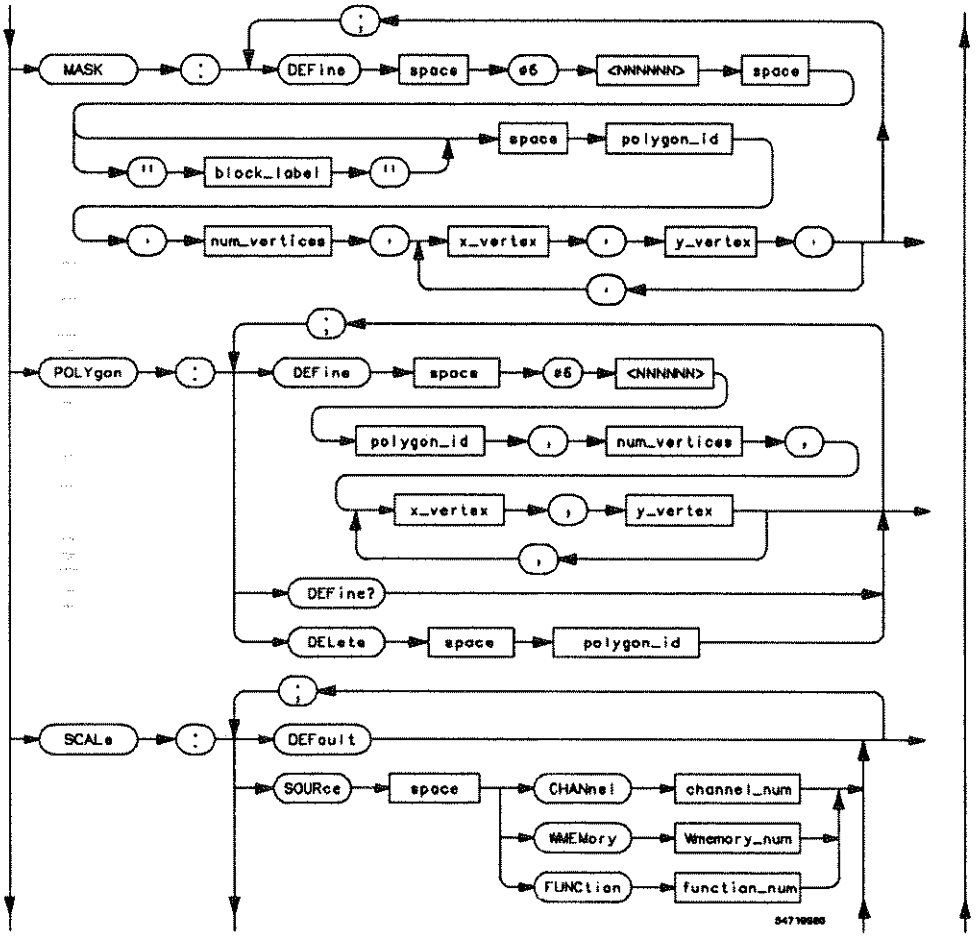
Mask Test Subsystem Functional Diagram

Figure 28-2



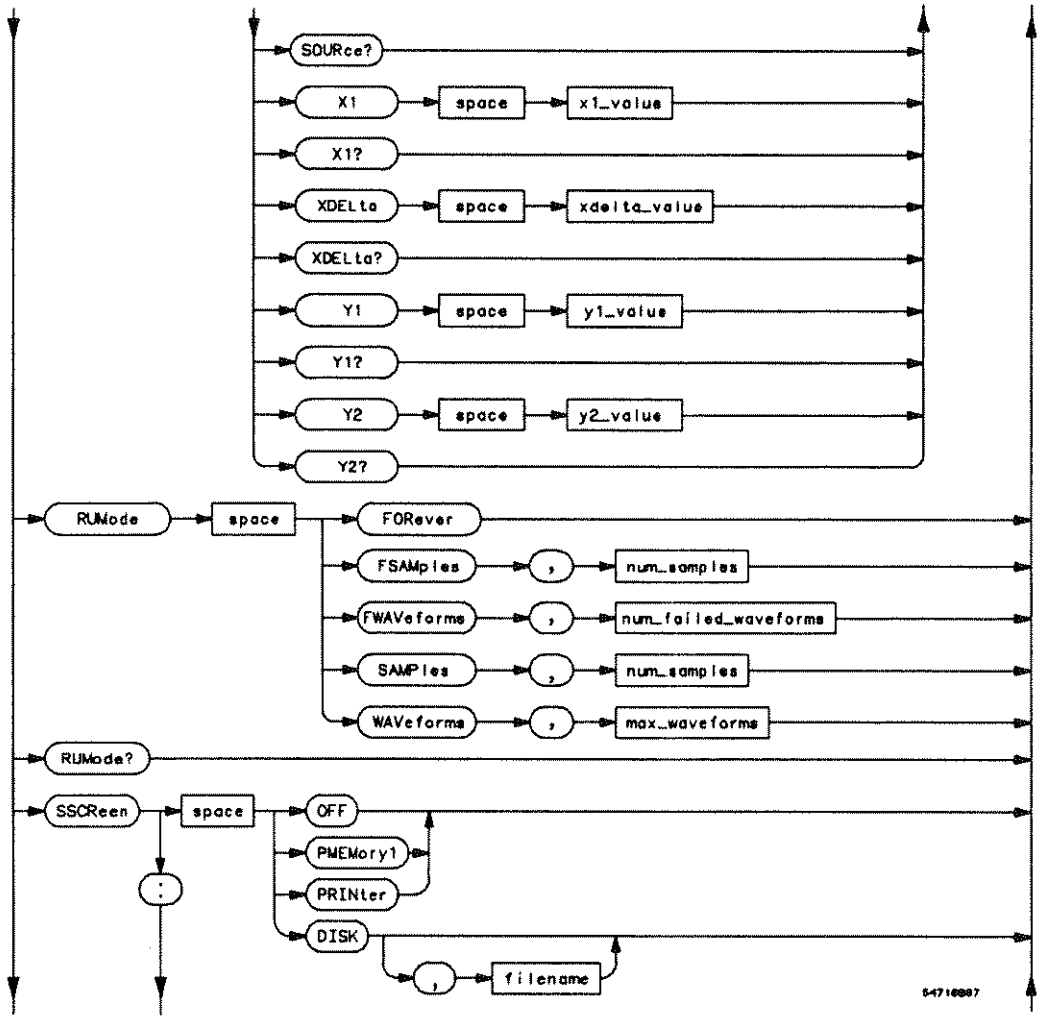
Mask Test Subsystem Syntax Diagram

Figure 28-2 (continued)



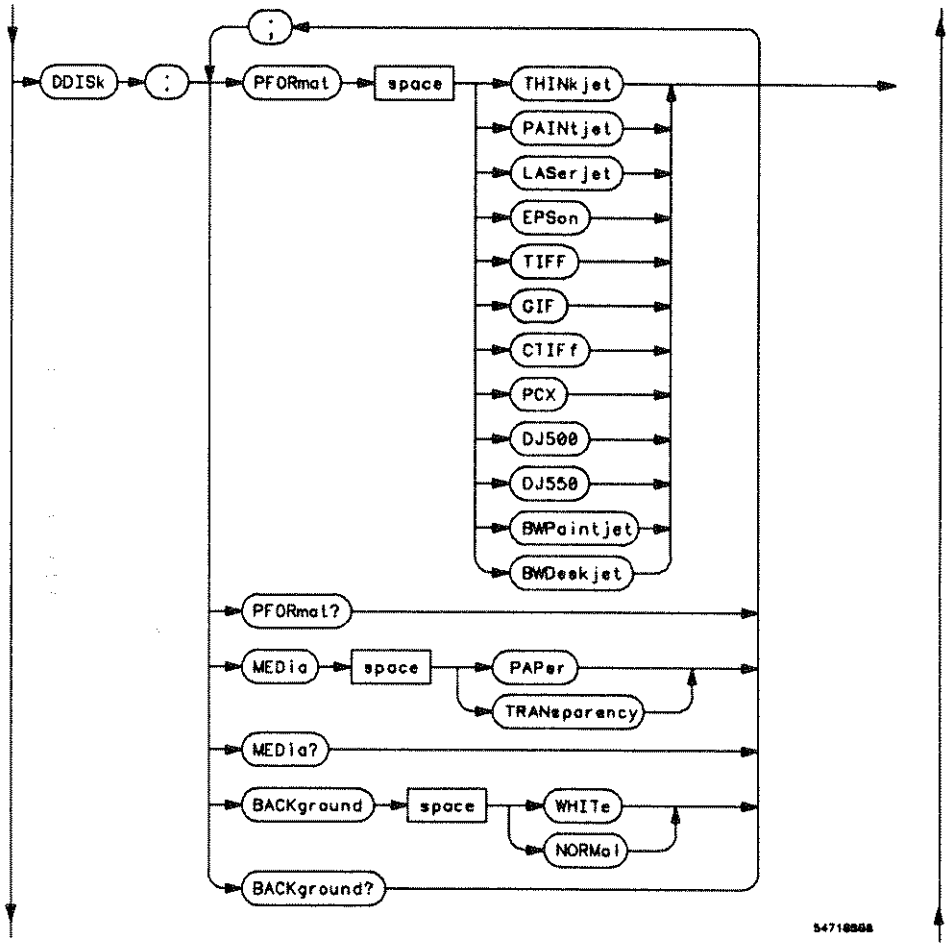
Mask Test Subsystem Syntax Diagram (continued)

Figure 28-2 (continued)



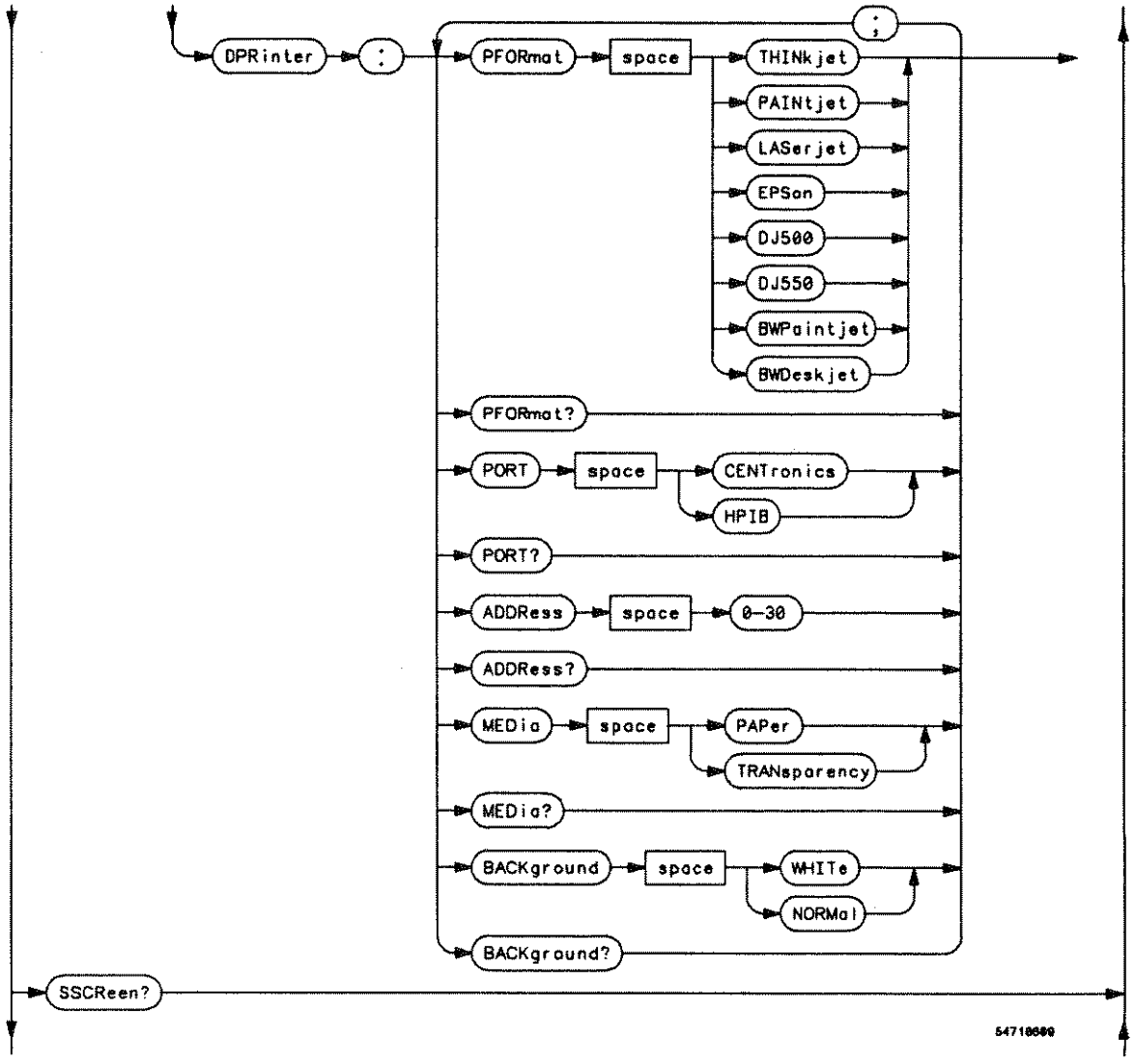
Mask Test Subsystem Syntax Diagram (continued)

Figure 28-2 (continued)



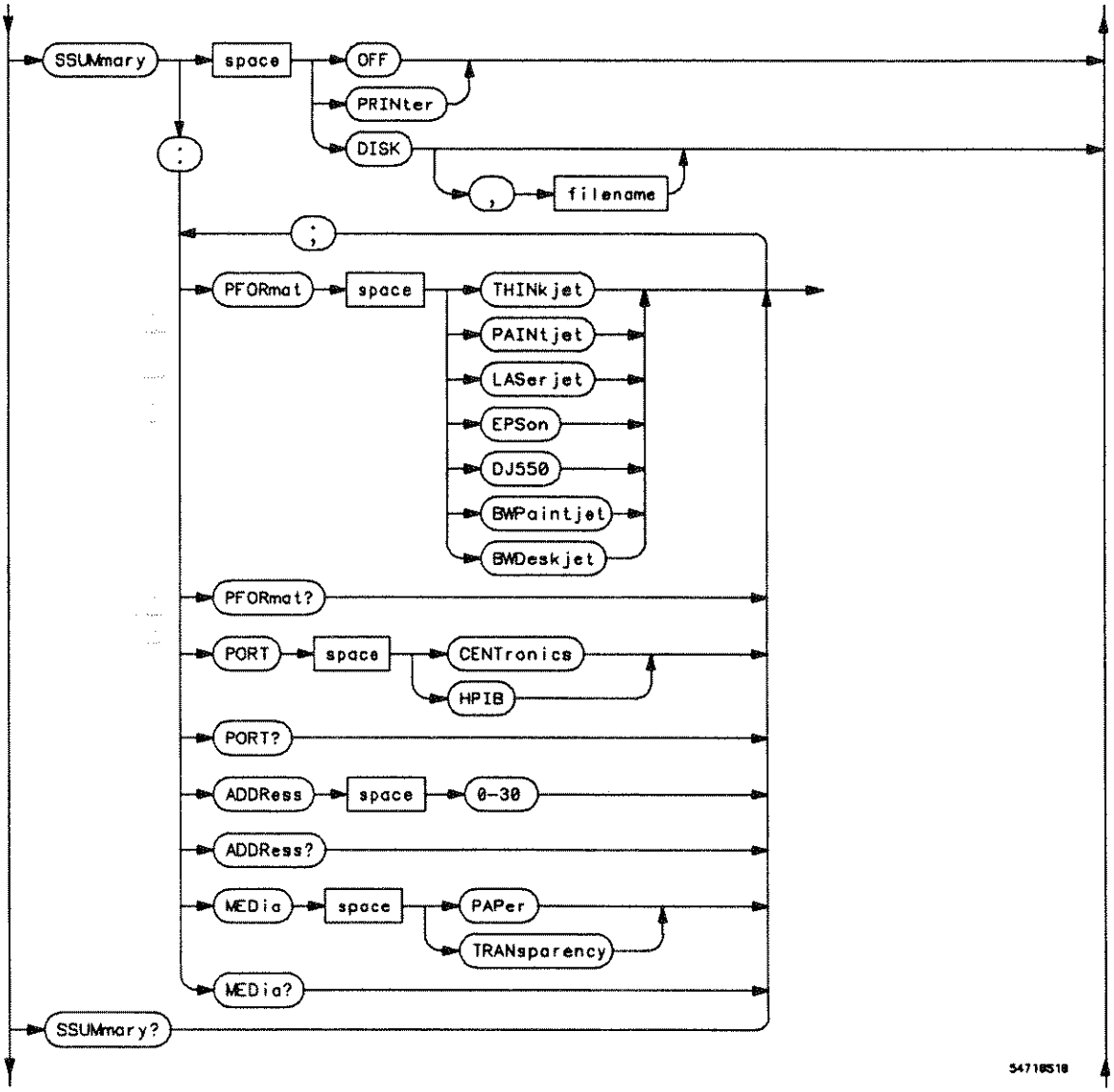
Mask Test Subsystem Syntax Diagram (continued)

Figure 28-2 (continued)



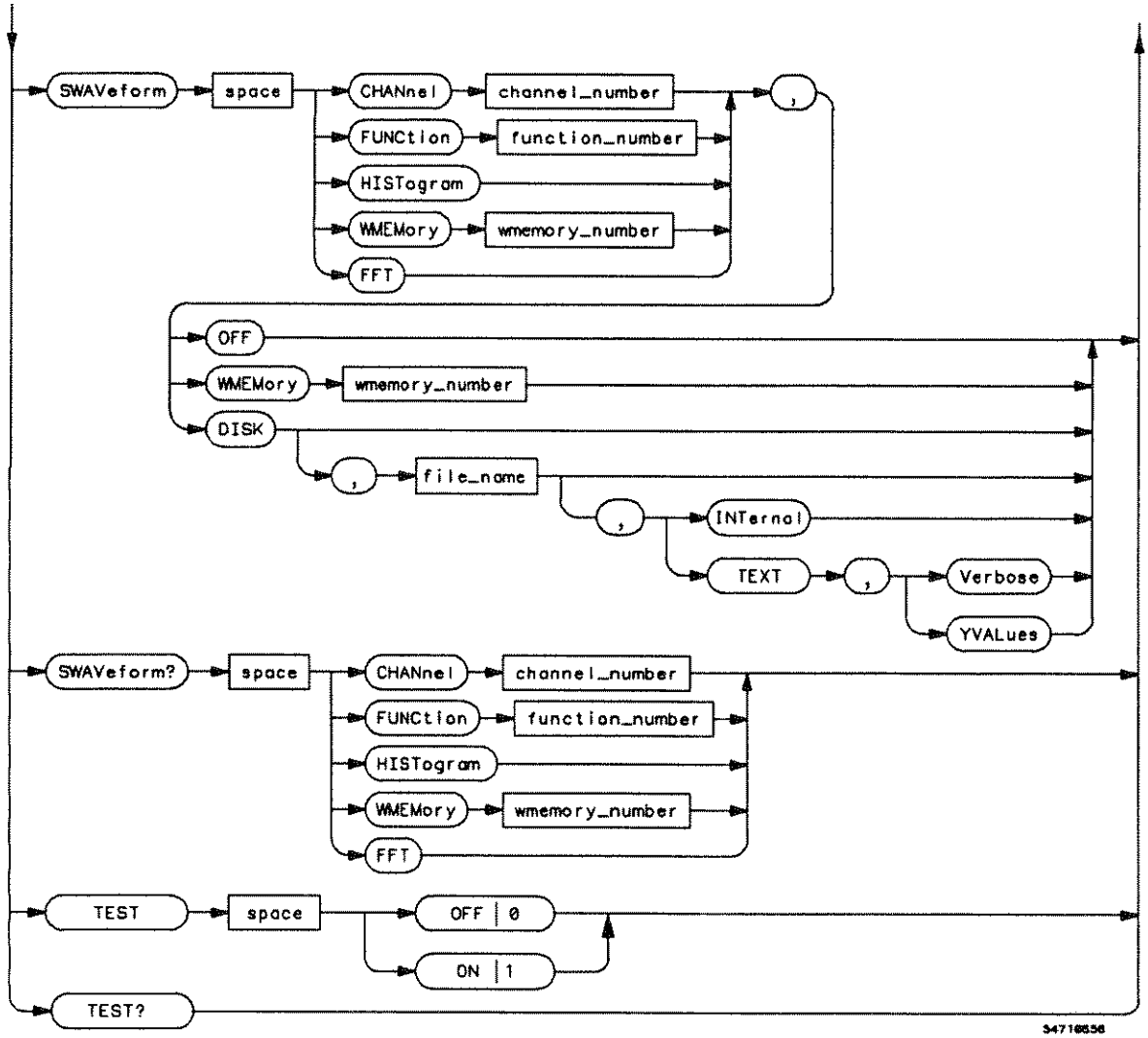
Mask Test Subsystem Syntax Diagram (continued)

Figure 28-2 (continued)



Mask Test Subsystem Syntax Diagram (continued)

Figure 28-2 (continued)



Mask Test Subsystem Syntax Diagram (continued)

AMASk:CReate

Command

```
:MTESt:AMASk:CReate
```

The AMASk:CReate command automatically constructs a mask around the current waveform on the display, using the tolerance parameters defined by the AMASk:XDELta, AMASk:YDELta, and AMASK:UNITs commands. The mask only encompasses the portion of the waveform visible on the display, so your measurement must ensure that the waveform is acquired and displayed consistently to obtain repeatable results.

Example

The following example defines an automask given the current ΔX, ΔY, and units settings.

```
10  OUTPUT 707;":MTEST:AMASK:CREATE"
20  END
```

AMASk:SOURce

Command

```
:MTESt:AMASk:SOURce {WMEMory<number> |
FUNCtion<number> | CHANnel<number> | FFT}
```

The :MTESt:AMASk:SOURce command selects the source for the interpretation of the AMASk:XDELta and AMASk:YDELta parameters when AMASk:UNITs is set to CURRent. When UNITs are CURRent, the XDELta and YDELta parameters are defined in terms of the measurement system of the selected source. Suppose that UNITs are CURRent and that you set SOURce to CHANNEL1, which is using volts as a measurement system. Then you can define AMASk:XDELta in terms of volts and AMASk:YDELta in terms of seconds.

`<number>` For channels: the number represents an integer, 1 through 4.

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following program sets the automask measurement system source to Channel 1.

```
10  OUTPUT 707;":MTEST:AMASK:SOURCE CHANNEL1"
20  END
```

Query

```
:MTESt:AMASk:SOURce?
```

The query returns the currently set source.

Returned Format

```
[:MTESt:AMASk:SOURce] {WMEMory<number> | FUNCtion<number> |
CHANnel<number> | FFT}<NL>
```

Example

The following program gets the source setting for automask and prints the result on the controller display.

```
10  DIM AMASK_SOURCE$[30]
20  OUTPUT 707;":MTEST:AMASK:SOURCE?"
30  ENTER 707;AMASK_SOURCE$
40  PRINT AMASK_SOURCE$
50  END
```

AMASk:UNITs

Command

```
:MTESt:AMASk:UNITs {CURRent | DIVisions}
```

The AMASk:UNITs command alters the way the mask test subsystem interprets the ΔX and ΔY tolerance parameters for automasking, defined by AMASk:XDELta and AMASk:YDELta respectively.

`CURRent` When set to CURRent, the mask test subsystem uses the measurement system currently in use, usually time for ΔX and voltage for ΔY.

`DIVisions` When set to DIVisions, the mask test subsystem uses the graticule as the measurement system, so that tolerance settings are specified as parts of a screen division.

The mask test subsystem maintains separate XDELta and YDELta settings for CURRent and DIVisions. Thus, XDELta and YDELta are not converted to new values when you change the UNITs setting.

Example

The following example sets the measurement units for automasking to the current measurement system.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS CURRENT"
20  END
```

Query

```
:MTESt:AMASk:UNITs?
```

The AMASk:UNITs query returns the current measurement units setting for the mask test automask feature.

Returned Format

```
[:MTESt:AMASk:UNITs] {CURRent | DIVision}<NL>
```

Example

The following example gets the current automask units setting, then prints the setting on the screen of the controller.

```
10  DIM AUTOMASK_UNITS$[10]
20  OUTPUT 707;":MTEST:AMASK:UNITS?"
30  ENTER 707;AUTOMASK_UNITS$
40  PRINT AUTOMASK_UNITS$
50  END
```

AMASk:XDELta

Command

```
:MTESt:AMASk:XDELta {<xdelta_value>}
```

The AMASk:XDELta command sets the tolerance in the X direction around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to X components of the waveform to determine the boundaries of the mask.

<xdelta_value> A value for the X tolerance. This value is interpreted based on the setting specified by the AMASk:UNITs command; thus, if you specify 250-E3, the setting for AMASk:UNITs is CURRent, and the current measurement system specifies time in the X direction, the tolerance will be ±250 ms. If the setting for AMASk:UNITs is DIVisions, the same xdelta_value will set the tolerance to ±250 millidivisions, or 1/4 of a division.

Example The following example sets the current measurement system to divisions and sets the ΔX tolerance to one-eighth of a division.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS DIVISIONS"
20  OUTPUT 707;":MTEST:AMASK:XDELTA 125E-3"
30  END
```

Query

```
:MTESt:AMASk:XDELta?
```

The AMASk:XDELta? query returns the current setting of the ΔX tolerance for automasking. If your controller program will interpret this value, it should also request the current measurement system using the AMASk:UNITs query.

Returned Format

```
[:MTESt:AMASk:XDELta] {<xdelta_value>}<NL>
```

Example

The following example gets the measurement system units and ΔX settings for automasking from the oscilloscope and prints the results on the controller screen.

```
10  DIM AUTOMASK_UNITS$[10]
20  DIM AUTOMASK_XDELTA$[20]
30  OUTPUT 707;":MTEST:AMASK:UNITS?"
40  ENTER 707;AUTOMASK_UNITS$
50  OUTPUT 707;":MTEST:AMASK:XDELTA?"
60  ENTER 707;AUTOMASK_XDELTA$
70  PRINT AUTOMASK_UNITS$
80  PRINT AUTOMASK_XDELTA$
90  END
```

AMASk:YDELta

Command

```
:MTESt:AMASk:YDELta {<ydelta_value>}
```

The AMASk:YDELta command sets the tolerance in the Y direction around the waveform for the automasking feature. The absolute value of the tolerance will be added and subtracted to Y components of the waveform to determine the boundaries of the mask.

<ydelta_value> A value for the Y tolerance. This value is interpreted based on the setting specified by the AMASk:UNITs command; thus, if you specify 250-E3, the setting for AMASk:UNITs is CURRent, and the current measurement system specifies voltage in the Y direction, the tolerance will be ±250 mV. If the setting for AMASk:UNITs is DIVisions, the same ydelta_value will set the tolerance to ±250 millidivisions, or 1/4 of a division.

Example The following example sets the current measurement system to current and sets the ΔY tolerance to 30 mV, assuming that the current measurement system specifies volts in the Y direction.

```
10  OUTPUT 707;":MTEST:AMASK:UNITS CURRENT"
20  OUTPUT 707;":MTEST:AMASK:YDELTA 30E-3"
30  END
```

Query

```
:MTESt:AMASk:YDELta?
```

The AMASk:YDELta? query returns the current setting of the ΔY tolerance for automasking. If your controller program will interpret this value, it should also request the current measurement system using the AMASk:UNITs query.

Returned Format

```
[:MTESt:AMASk:YDELta] {<ydelta_value>}<NL>
```

Example

The following example gets the measurement system units and ΔY settings for automasking from the oscilloscope and prints the results on the controller screen.

```
10  DIM AUTOMASK_UNITS$[10]
20  DIM AUTOMASK_YDELTA$[20]
30  OUTPUT 707;":MTEST:AMASK:UNITS?"
40  ENTER 707;AUTOMASK_UNITS$
50  OUTPUT 707;":MTEST:AMASK:YDELTA?"
60  ENTER 707;AUTOMASK_YDELTA$
70  PRINT AUTOMASK_UNITS$
80  PRINT AUTOMASK_YDELTA$
90  END
```

COUNt:FAILures?

Query

```
:MTESt:COUNt:FAILures? {POLYgon<number>}
```

The MTESt:COUNt:FAILures? query returns the number of failures that occurred within a particular polygon. By defining polygons within polygons, then counting the failures for each individual polygon, you can implement testing at different tolerance levels for a given waveform.

The value 9.999E37 is returned if mask testing is not enabled or if you specify a polygon number that is not used.

`<number>` An integer, 1 through 8, designating the polygon for which you want to determine the failure count.

Returned Format

```
[:MTESt:COUNt:FAILures] POLYgon<number>
<number_of_failures>}<NL>
```

`<number_of_failures>` The number of failures that have occurred for the designated polygon.

Example

The following example determines the current failure count for polygon 3 and prints it on the controller screen.

```
10  DIM MASK_FAILURES$[50]
20  OUTPUT 707;":MTEST:COUNT:FAILURES? POLYGON3"
30  ENTER 707;MASK_FAILURES$
40  PRINT MASK_FAILURES$
50  END
```

COUNt:FSAMples?

Query

```
:MTESt:COUNt:FSAMples?
```

The MTESt:COUNt:FSAMples? query returns the total number of failed samples in the current mask test run. This count is for all polygons and all waveforms, so if you wish to determine failures by polygon number, use the COUNt:FAILures? query.

The count value returned is not the sum of the failure counts for each polygon. For example, assume a polygon 2 enclosed completely by polygon 1. If polygon 1 has 100 failures, the value returned is 100, regardless of how many failures are in polygon 2. Because polygon 2 is completely enclosed, the failure count for polygon 2 must be less than or equal to 100 in this instance.

The value 9.999E37 is returned if mask testing is not enabled.

Returned Format

```
[:MTESt:COUNt:FSAMples] {<number_of_failed_samples>}<NL>
```

`<number_of_failed_samples>` The total number of failed samples for the current test run.

Example

The following example determines the number of failed samples and prints the result on the controller screen.

```
10  DIM MASK_FSAMPLES$[50]
20  OUTPUT 707;":MTEST:COUNT:FSAMPLES?"
30  ENTER 707;MASK_FSAMPLES
40  PRINT MASK_FSAMPLES$
50  END
```

COUNt:FWAVeforms?

Query

```
:MTESt:COUNt:FWAVeforms?
```

The MTESt:COUNt:FWAVeforms? query returns the total number of failed waveforms in the current mask test run. This count is for all polygons and all waveforms, so if you wish to determine failures by polygon number, use the COUNt:FAILures? query.

This count may not always be available. It is available only when the following conditions are true:

- Mask testing was turned on before the histogram or color-graded display, and
- No mask changes have occurred, including scaling changes, editing, or new masks.

The value 9.999E37 is returned if mask testing is not enabled, or if you have modified the mask.

Returned Format

```
[:MTESt:COUNt:FWAVeforms] {<number_of_failed_waveforms>}<NL>
```

`<number_of_failed_waveforms>` The total number of failed waveforms for the current test run.

Example

The following example determines the number of failed waveforms and prints the result on the controller screen.

```
10  DIM MASK_FWAVEFORMS$[50]
20  OUTPUT 707;":MTEST:COUNT:FWAVEFORMS?"
30  ENTER 707;MASK_FWAVEFORMS
40  PRINT MASK_FWAVEFORMS$
50  END
```

COUNt:SAMPles?

Query

```
:MTESt:COUNt:SAMPles?
```

The MTESt:COUNt:SAMPles? query returns the total number of samples captured in the current mask test run.

The value 9.999E37 is returned if mask testing is not enabled.

Returned Format

```
[:MTESt:COUNt:SAMPles] {<number_of_samples>}<NL>
```

`<number_of_samples>` The total number of samples for the current test run.

Example

The following example determines the number of samples gathered in the current test run and prints the result on the controller screen.

```
10  DIM MASK_SAMPLES$[50]
20  OUTPUT 707;":MTEST:COUNT:SAMPLES?"
30  ENTER 707;MASK_SAMPLES
40  PRINT MASK_SAMPLES$
50  END
```

COUNt:WAVeforms?

Query

```
:MTESt:COUNt:WAVeforms?
```

The MTESt:COUNt:WAVeforms? query returns the total number of waveforms gathered in the current mask test run.

The value 9.999E37 is returned if mask testing is not enabled.

Returned Format

```
[:MTESt:COUNt:WAVeforms] {<number_of_waveforms>}<NL>
```

`<number_of_waveforms>` The total number of waveforms for the current test run.

Example

The following example determines the number of waveforms gathered in the current test run and prints the result on the controller screen.

```
10  DIM MASK_WAVEFORMS$[50]
20  OUTPUT 707;":MTEST:COUNT:WAVEFORMS?"
30  ENTER 707;MASK_WAVEFORMS
40  PRINT MASK_WAVEFORMS$
50  END
```

MASK:DEFine

Command

```
:MTESt:MASK:DEFine #6NNNNNN [<mask_label>]
{<polygon_id>,<number_of_vertices>,
<x_coordinate>,<y_coordinate>
[,<x_coordinate>,<y_coordinate>]*}
{,<polygon_id>,<number_of_vertices>,
<x_coordinate>, <y_coordinate>
[,<x_coordinate>,<y_coordinate>]*}*
```

The MTESt:MASK:DEFine command defines a set of up to eight polygons to be used for mask testing. When you use the MASK:DEFine command, you can specify polygon definitions for up to eight polygons. You can also use the MTESt:POLYgon:DEFine command to define a single polygon at a time.

The set can be named with an optional mask label. Each polygon in the mask is described by a polygon identifier and a series of vertices, specified as X-Y coordinates. An initial block identifier, starting with #6 and followed immediately by a six-digit integer, specifies the number of bytes of polygon data to follow. You may include "C"-style comments in the polygon data; these comments are enclosed with /* and */. You must include the byte count for the comments in the byte count supplied with the command.

Items in the mask definition must be separated by spaces. As used here, spaces are defined to be character space (decimal 32 ASCII), commas, tabs, or newline characters.

#6NNNNNN The #6 indicates that the immediately following 6-digit integer, represented by NNNNNN, will contain the byte count for the mask data.

<mask_label> An optional string naming this mask definition.

<polygon_id> An integer, 1 through 8, identifying the polygon you wish to define.

<number_of_vertices> The number of vertices that you will supply as X-Y coordinates.

<x_coordinate> A value specifying the x coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

`<y_coordinate>` A value specifying the y coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

Example

The following example shows how to define polygons 1 and 2 as rectangular polygons below and above the Y1 and Y2 boundaries.

```
10  OUTPUT 707 USING "-K";":MTEST:MASK:DEFINE #6000048
1,4,0,MIN,1,MIN,1,0,0,0,2,4,0,1,1,1,1,MAX,0,MAX"
20  END
```

The "-K" is an image specifier that outputs the string in compact form with no leading or trailing blanks.

Query

```
:MTESt:MASK:DEFine?
```

The MASk:DEFine? query gets the parameters for all currently defined polygons from the oscilloscope.

Returned Format

```
[:MTESt:MASK:DEFine] #6NNNNNN [<mask_label>]
{<polygon_id>,<number_of_vertices>,
<x_coordinate>,<y_coordinate>
[,<x_coordinate>,<y_coordinate>]*}
{,<polygon_id>, <number_of_vertices>,
<x_coordinate>,<y_coordinate>
[,<x_coordinate>,<y_coordinate>]*}*<NL>
```

Example

This example gets the definition of the mask and prints it on the controller screen.

```
10  DIM MASK_DEF$[400]
20  OUTPUT 707;":MTEST:MASK:DEFINE?"
30  ENTER 707 USING "-K";POLY_DEF$
40  PRINT MASK_DEF$
50  END
```

The "-K" in the ENTER statement is an image specifier that forces entered characters to be placed in the string. Carriage returns without line-feeds are also placed in the string without terminating the entry.

POLYgon:DEFine

Command

```
:MTESt:POLYgon:DEFine #6NNNNNN <polygon_id>,
<number_of_vertices>,<x_coordinate>,<y_coordinate>
{[,<x_coordinate>,<y_coordinate>]*}
```

The MTESt:POLYgon:DEFine command defines a polygon to be used for mask testing. You can define one polygon at a time by using this command. If you want to define several polygons at once, you can use the DISK:LOAD command to load a mask file, or use the MTESt:MASK:DEFine command to define a series of polygons.

Each polygon is described by a polygon identifier and a series of vertices, specified as X-Y coordinates. An initial block identifier, starting with #6 and followed immediately by a six-digit integer, specifies the number of bytes of polygon data to follow. You may include "C"-style comments in the polygon data; these comments are enclosed with /* and */. You must include the byte count for the comments in the byte count supplied with the command.

Items in the polygon definition must be separated by spaces. As used here, spaces are defined to be character space (decimal 32 ASCII), commas, tabs, or newline characters.

#6NNNNNN The #6 indicates that the immediately following 6-digit integer, represented by NNNNNN, will contain the byte count for the polygon data.

<polygon_id> An integer, 1 through 8, identifying the polygon you wish to define.

<number_of_vertices> The number of vertices that you will supply as X-Y coordinates.

<x_coordinate> A value specifying the x coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

<y_coordinate> A value specifying the y coordinate of a polygon vertex relative to the scaled coordinate system. See the MTESt:SCALe commands. The values MIN and MAX automatically set the vertex to the boundary of the graticule, regardless of the scaling system in use.

Example

The following example shows how to define polygon 3 as a triangular polygon.

```
10  OUTPUT 707 USING "-K";":MTEST:POLYGON:DEFINE #6000028
3,3,1.0,1.0,0.5,0.7,0.7,0.7"
20  END
```

The "-K" is an image specifier that outputs the string in compact form with no leading or trailing blanks.

Query

`:MTESt:POLYgon:DEFine? POLYGON<polygon_number>`

The POLYgon:DEFine? query gets the parameters for the selected polygon from the oscilloscope. To get the parameters for all defined polygons, use the :MTESt:MASK:DEFine? query.

`<polygon_number>` A number, 1 through 8, specifying the polygon for which you want information.

Returned Format

```
[:MTESt:POLYgon:DEFine] #6NNNNNN
<polygon_id>,<number_of_vertices>,<x_coordinate>,<y_coordinate>
{[,<x_coordinate>,<y_coordinate>]*}<NL>
```

Example

This example gets the definition of polygon one and prints it on the controller screen.

```
10  DIM POLY_DEF$[200]
20  OUTPUT 707;":MTEST:POLYGON:DEFINE? POLYGON1"
30  ENTER 707 USING "-K";POLY_DEF$
40  PRINT POLY_DEF$
50  END
```

The "-K" in the ENTER statement is an image specifier that forces entered characters to be placed in the string. Carriage returns without line-feeds are also placed in the string without terminating the entry.

RUMode

Command

```
:MTESt:RUMode {FORever |
FSAMples, <number_of_failed_samples> |
FWAVeforms, <number_of_failed_waveforms> |
SAMPles, <number_of_samples> |
WAVeforms, <number_of_waveforms>}
```

The :MTESt:RUMode (RunUntilMode) command determines the termination conditions for the test. The choices are FORever, FSAMples (FailedSAMples), FWAVeforms (FailedWAVeforms), SAMPles, or WAVeforms.

If FSAMples, FWAVeforms, SAMPles, or WAVeforms are selected, a second parameter is required indicating the number of failures that can occur or the number of samples or waveforms that are to be acquired.

FORever FORever runs the Mask Test until the test is turned off. This is used when you want a measurement to run continually and to not stop after a fixed number of failures. For example, you may want the Mask Test to run overnight and not be limited by a number of failures.

FSAMples FSAMples runs the Mask Test until at least a set number of failed samples occur, that is, samples which fell within the boundaries of one or more polygons. When FSAMples is sent, the test executes until the selected total failures are obtained. After each complete acquisition, the number of failures are compared against this number to test for termination.

<number_of_failed_samples> An integer: 1 to 2,000,000,000.

FWAVeforms FWAVeforms runs the Mask Test until a set number of failed waveforms occur, that is, waveforms which fell within the boundaries of one or more polygons. When FWAVeforms is sent, the test executes until the selected total failures are obtained. The number of failures are compared against this number to test for termination.

<number_of_failed_waveforms> An integer: 1 to 1,000,000,000.

Use the FSAMples or FWAVeforms mode when you want the Mask Test to complete after a set number of failures.

SAMPles SAMPles sets the minimum number of samples that are required to terminate the test. After each acquisition, the number of samples acquired is compared against this value. SAMPles runs the Mask Test until at least a set number of samples are acquired.

<number_of_samples> An integer, 1 to 2,000,000,000.

WAVeforms WAVeforms sets the maximum waveforms that are required. When any measurement reaches this number of waveforms the test terminates. WAVeforms runs the Mask Test until a set number of waveforms are acquired.

In the real-time acquisition mode, a waveform is acquired with each trigger event. That is because in the real-time acquisition mode, all the data points that make up a waveform are acquired from a single trigger event. If you select 10 waveforms, that is the same as 10 trigger events.

In the equivalent-time sampling mode, a waveform is reconstructed from several trigger events. The Mask Test operates in conjunction with the ACQuire:COMPlete command. If the ACQuire:COMPlete is set to OFF, measurements and Mask Test consider each trigger event as a waveform. Data from previous triggers is still retained in memory and the latest trigger may or may not add additional data points to the waveform.

When ACQuire:COMPlete is ON, the command acts like a holdoff control because the measurement is held off until after the completion criteria is met. If Mask Test is ON, it also waits for a measurement before testing the measurement results. For example, if completion is set to 95%, then measurements are performed on the data after 95% of the waveform record is filled with data. Depending on the setup of the oscilloscope, it can take several triggers before the completion criteria is met. After the completion criteria is met, measurements are performed and Mask Test checks the measurement results. The data is then cleared from memory, and the oscilloscope starts acquiring new data.

> Complete percentage is set with the ACQuire:COMPlete command.

Use the SAMPles or WAVeforms mode when you want the Mask Test to reach completion after a set number of samples waveforms are acquired. The

test terminates when the system reaches the specified number of samples or waveforms.

`<number_of_waveforms>` An integer: 1 to 1,000,000,000.

Example

The following example sets the mask test subsytem run-until mode to continue testing until 500,000 samples have been gathered.

```
10  OUTPUT 707;":MTEST:RUMODE SAMPLES,500E3"
20  END
```

Query

`:MTESt:RUMode?`

The query returns the currently selected termination condition and value.

Returned Format

```
[:MTESt:RUMode] {FOREver |
FSAMples, <number_of_failed_samples> |
FWAVeforms, <number_of_failed_waveforms> |
SAMPles, <number_of_samples> |
WAVeforms, <number_of_waveforms>}
```

Example

The following example gets the current setting of the mask test run-until mode from the oscilloscope and prints it on the controller screen.

```
10  DIM MTEST_RUMODE$[50]
20  OUTPUT 707;":MTEST:RUMODE?"
30  ENTER 707;MTEST_RUMODE$
40  PRINT MTEST_RUMODE$
50  END
```

SCALe:DEFault

Command

```
:MTESt:SCALe:DEFault
```

The MTESt:SCALe:DEFault commands sets the scaling markers to default values. The X1, XDELta, Y1, and Y2 markers are set to values corresponding to graticule positions that are one division in from the left, right, top, and bottom of the graticule respectively.

Example

The following example selects the default scale.

```
10  OUTPUT 707;":MTEST:SCALE:DEFAULT"
20  END
```

SCALe:SOURce

Command

```
:MTESt:SCALe:SOURce {WMEMory<number> |
FUNCtion<number> | CHANnel<number> }
```

The SCALe:SOURce command sets the source used by the mask subsystem for interpretation of the SCALe:Y1 and SCALe:Y2 parameters. SCALe:Y1 and SCALe:Y2 set the vertical boundaries of the coordinate system for mask testing, and are affected by the scaling of the selected source. For example, suppose that Y1 was set to -1 V and Y2 was set to +1 V. If Channel 1 was selected as the scaling source, and was set to a vertical scale factor of 100 mV per division, the Y1 and Y2 markers will be below and above the graticule, respectively. If Channel 2 was selected as the scaling source, and was set to a vertical scale factor of 500 mV per division, the Y1 and Y2 markers will be two divisions below and above the center of the graticule, respectively.

Interpretation of the X1 and ΔX settings is done using the time/div setting in the time base subsystem, if the source is a channel. The setting can be queried using the TIMEbase:SCALe? command. Functions and waveform memories can have independent scale.

`<number>` For channels: the number represents an integer, 1 through 4.

For waveform memories: 1, 2, 3, or 4.

For functions: 1 or 2.

Example

The following example selects wavoform memory 1 as the source for interpretation of the Y1 and Y2 scaling values.

```
10  OUTPUT 707;":MTEST:SCALE:SOURCE WMEMORY1"
20  END
```

Query

```
:MTESt:SCALe:SOURce?
```

The SCALe:SOURce? query returns the name of the source currently used to interpret the Y1 and Y2 scale factors.

Returned Format

```
[:MTESt:SCALe:SOURce] {WMEMory<number> | FUNCtion<number> |
CHANnel<number> }<NL>
```

Example

The following example gets the current scale source setting from the oscilloscope and prints it on the controller screen.

```
10  DIM SCALE_SOURCE$[30]
20  OUTPUT 707;":MTEST:SCALE:SOURCE?"
30  ENTER 707;SCALE_SOURCE$
40  PRINT SCALE_SOURCE$
50  END
```

SCALe:X1

Command

```
:MTESt:SCALe:X1 {<x1_value>}
```

The SCALe:X1 command defines where X=0 in the base coordinate system used for mask testing. The other X-coordinate is defined by the SCALe:XDELta command. Once the X1 and XDELta coordinates are set, all X values of vertices in polygon masks are defined with respect to this value, according to the equation:

$X_{actual} = (X_{vertex} \times \Delta X) + X1$

Thus, if you set X1 to 100 μs, and XDELta to 100 μs, an X value of .100 in a vertex is at 110 μs.

The oscilloscope uses this equation to normalize vertex values. This simplifies reprogramming to handle different data rates. For example, if you halve the period of the waveform of interest, you need only to adjust the XDELta value to set up the mask for the new waveform.

`<x1_value>` A time value specifying the location of the X1 coordinate, which will then be treated as X=0 for polygon vertex coordinates.

Example

The following example sets the X1 coordinate at 150 ms.

```
10  OUTPUT 707;":MTEST:SCALE:X1 150E-3"
20  END
```

Query

```
:MTESt:SCALe:X1?
```

The SCALe:X1? query returns the current X1 coordinate setting.

Returned Format

```
[:MTESt:SCALe:X1] {<x1_value>}<NL>
```

Example

The following example gets the current setting of the X1 coordinate from the oscilloscope and prints it on the controller screen.

```
10  DIM SCALE_X1$[50]
20 OUTPUT 707;":MTEST:SCALE:X1?"
30 ENTER 707;SCALE_X1$
40 PRINT SCALE_X1$
50 END
```

SCALe:XDELta

Command

```
:MTESt:SCALe:XDELta {<xdelta_value>}
```

The SCALe:XDELta command defines the position of the X2 marker with respect to the X1 marker. In the mask test coordinate system, the X1 marker defines where X=0; thus, the X2 marker defines where X=1. Because all X vertices of polygons defined for mask testing are normalized with respect to X1 and ΔX, redefining ΔX also moves those vertices to stay in the same locations with respect to X1 and ΔX. Thus, in many applications, it is best if you define XDELta as a pulse width or bit period. Then a change in data rate without corresponding changes in the waveform can easily be handled by changing ΔX.

The X-coordinate of polygon vertices are normalized using the equation:

$$X_{actual} = (X_{vertex} \times \Delta X) + X1$$

`<xdelta_value>` A time value specifying the distance of the X2 marker with respect to the X1 marker.

Example

Assume that the period of the waveform you wish to test is 1 μs. Then the following example will set ΔX to 1 μs, ensuring that the waveform's period is between the X1 and X2 markers.

```
10  OUTPUT 707;":MTEST:SCALE:XDELTA 1E-6"
20  END
```

Query

```
:MTESt:SCALe:XDELta?
```

The SCALe:XDELta? query returns the current value of ΔX.

Returned Format

```
[:MTESt:SCALe:XDELta] {<xdelta_value>}<NL>
```

Example

The following example gets the value of ΔX from the oscilloscope and prints it on the controller screen.

```
10  DIM SCALE_XDELTA$[50]
20  OUTPUT 707;":MTEST:SCALE:XDELTA?"
30  ENTER 707;SCALE_XDELTA$
40  PRINT SCALE_XDELTA$
50  END
```

SCALe:Y1

Command

```
:MTESt:SCALe:Y1 {<y1_value>}
```

The SCALe:Y1 command defines where Y=0 in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries set by SCALe:Y1 and SCALE:Y2 according to the equation:

$$Y_{actual} = (Y_{vertex} \times (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1 V, a Y value of .100 in a vertex is at 190 mV.

`<y1_value>` A voltage value specifying the point at which Y=0.

Example

The following example sets the Y1 marker to -150 mV.

```
10  OUTPUT 707;":MTEST:SCALE:Y1 -150E-3"
20  END
```

Query

```
:MTESt:SCALe:Y1?
```

The SCALe:Y1? query returns the current setting of the Y1 marker.

Returned Format

```
[:MTESt:SCALe:Y1] {<y1_value>}<NL>
```

Example

The following example gets the setting of the Y1 marker from the oscilloscope and prints it on the controller screen.

```
10  DIM SCALE_Y1$[50]
20  OUTPUT 707;":MTEST:SCALE:Y1?"
30  ENTER 707;SCALE_Y1$
40  PRINT SCALE_Y1$
50  END
```

SCALe:Y2

Command

```
:MTESt:SCALe:Y2 {<y2_value>}
```

The SCALe:Y2 command defines the Y2 marker in the coordinate system for mask testing. All Y values of vertices in the coordinate system are defined with respect to the boundaries defined by SCALe:Y1 and SCALE:Y2 according to the following equation:

$$Y_{actual} = (Y_{vertex} \times (Y2 - Y1)) + Y1$$

Thus, if you set Y1 to 100 mV, and Y2 to 1 V, a Y value of .100 in a vertex is at 190 mV.

`<y2_value>` A voltage value specifying the location of the Y2 marker.

Example

The following example sets the Y2 marker to 2.5 V.

```
10  OUTPUT 707;":MTEST:SCALE:Y2 2.5"
20  END
```

Query

```
:MTESt:SCALe:Y2?
```

The SCALe:Y2? query returns the current setting of the Y2 marker.

Returned Format

```
[:MTESt:SCALe:Y2] {<y2_value>}<NL>
```

Example

The following example gets the setting of the Y2 marker from the oscilloscope and prints it on the controller screen.

```
10  DIM SCALE_Y2$[50]
20  OUTPUT 707;":MTEST:SCALE:Y2?"
30  ENTER 707;SCALE_Y2$
40  PRINT SCALE_Y2$
50  END
```

SSCReen

Command

```
:MTESt:SSCReen {OFF | PMEMory1 | PRINter | DISK
[,<filename>]}
```

The :MTESt:SSCReen (Store SCReen) command saves a copy of the screen in the event of a failure.

A destination of OFF turns off the save action.

A destination of PMEMory1 adds the screen to the pixel memory.

When the destination is PRINter, a variety of printer-related controls are used to specify the printer. When the destination is DISK, a different set of commands is provided to control the print to disk. The printer specifications are set through the DPRinter (Destination Printer) and the DDISk (Destination Disk) commands.

`<filename>` A file prefix of four characters. If no prefix is specified, the default prefix is SCRN.

Example

The following program sends the screen image to the printer when a failure occurs.

```
10  OUTPUT 707;":MTEST:SSCREEN:PRINTER"
20  END
```

Query

```
:MTESt:SSCReen?
```

The query returns the current state of the SSCreen command.

Returned Format

```
[:MTESt:SSCReen]{OFF | PMEMory1 | PRINter | DISK
[,<filename>]} <NL>
```

Example

The following program gets the current destination for the save screen command and prints it on the controller.

```
10  DIM SCREEN_OUT$[30]
20  OUTPUT 707;":MTEST:SSCREEN?"
30  ENTER 707;SCREEN_OUT$
40  PRINT SCREEN_OUT$
50  END
```

The MTESt:SSCReen command contains the following command subsystems used for handling printer or disk output format:

- DDISk (Display Disk), and
- DPRinter (Display Printer).

SSCReen:DDISk

The MTESt:SSCReen:DDISk sub-subsystem commands are used to set up the disk drive when storing the Mask Test display screen to a disk. The disk drive setup consists of the following commands:

- BACKground,
- MEDia, and
- PFORmat (printer format).

SSCReen:DDISk:BACKground

Command

```
:MTESt:SSCReen:DDISk:BACKground {WHITe | NORMal}
```

This command controls the background color of the graticule area of an HP PaintJet print. It is valid only when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the disk.

Example

The following program selects the HP PaintJet as the printer format and sets the background color to white.

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT PAINTJET"
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:BACKGROUND WHITE"
30  END
```

Query

```
:MTESt:SSCReen:DDISk:BACKground?
```

The query returns the current background for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:BACKground] {WHITe | NORMal}<NL>
```

Example

The following example gets the current background color setting and prints the setting on the controller display.

```
10  DIM SCREEN_BACK$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:BACKGROUND?"
30  ENTER 707;SCREEN_BACK$
40  PRINT SCREEN_BACK$
50  END
```

SSCReen:DDISk:MEDia

Command

```
:MTESt:SSCReen:DDISk:MEDia {PAPer | TRANsparency }
```

The MEDia command specifies whether paper or transparency is to be used in the printer. This control is used when the store screen is directed to the disk. If TRANsparency is selected, the printer makes two passes over each data row, putting more ink on the page. This darkens the page for better transparency results. The command applies only to the HP PaintJet and Color DeskJet printers, and will slow print speed.

Example

The following program sets the media type to TRANsparency.

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:MEDIA TRANSPARENCY"
20  END
```

Query

```
:MTESt:SSCReen:DDISk:MEDia?
```

The query returns the current media for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:MEDia] {PAPer | TRANsparency }<NL>
```

Example

The following program gets the current media setting and prints the result on the controller screen.

```
10  DIM SCREEN_MEDIA$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:MEDIA?"
30  ENTER 707;SCREEN_MEDIA$
40  PRINT SCREEN_MEDIA$
50  END
```

SSCReen:DDISk:PFORmat

Command

```
:MTESt:SSCReen:DDISk:PFORmat { THINkjet | PAINtjet
| LASerjet | EPSon | GIF | TIFF | CTIFf | PCX |
DJ500 | DJ550 | BWPaintjet | BWDeskjet}
```

The PFORmat (Print FORmat) command selects the printer format to use when storing the screen to a disk. This includes all print formats supported by the instrument.

Example

The following example sets the output format for the save screen disk file to TIFF (Tag Image File Format).

```
10  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT TIFF"
20  END
```

Query

```
:MTESt:SSCReen:DDISk:PFORmat?
```

The query returns the current printer format for the disk store.

Returned Format

```
[:MTESt:SSCReen:DDISk:PFORmat] { THINkjet | PAINtjet |
LASerjet | EPSon | GIF | TIFF | CTIFf | PCX | DJ500 | DJ550 |
BWPaintjet | BWDeskjet}<NL>
```

Example

The following program gets the current setting for the disk file output format and prints the result on the controller screen.

```
10  DIM DISK_FORMAT$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DDISK:PFORMAT?"
30  ENTER 707;DISK_FORMAT$
40  PRINT DISK_FORMAT$
50  END
```

SSCReen:DPRinter

The MTESt:SSCReen:DPRinter sub-subsystem commands are used to set up the printer when storing the Mask Test display screen to a printer. The printer setup consists of the following commands:

- ADDRess,
- BACKground,
- MEDia,
- PFORmat, and
- PORT.

SSCReen:DPRinter:ADDRess

Command

```
:MTESt:SSCReen:DPRinter:ADDRess <address_value>
```

The ADDRess command allows the user to select the HP-IB address for the printer. This address is used only if the port is HP-IB.

`<address_value>` Any HP-IB address 0-30.

Example

The following example sets the port to HP-IB for save screen printer output, and tells the oscilloscope that the printer is at HP-IB address 8.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT HPIB"
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:ADDRESS 8"
30  END
```

Query

```
:MTESt:SSCReen:DPRinter:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:MTESt:SSCReen:DPRint:ADDRess] <address_value><NL>
```

Example

The following example gets the HP-IB address of the printer from the oscilloscope and prints the result on the controller screen.

```
10  DIM SCREEN_ADDRESS$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:ADDRESS?"
30  ENTER 707;SCREEN_ADDRESS$
40  PRINT SCREEN_ADDRESS$
50  END
```

SSCReen:DPRinter:BACKground

Command

```
:MTESt:SSCReen:DPRinter:BACKground {WHITe | NORMal}
```

This command controls the background color of the graticule area of an HP PaintJet print. It is only valid when the print format is an HP PaintJet. In NORMal, the selected screen color is used for that area. In WHITe, the background area is forced to white (the color of the printer paper). This control is used when the store screen is directed to the printer.

Example

The following program selects the HP PaintJet as the printer format and sets the background color to white.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT PAINTJET"
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:BACKGROUND WHITE"
30  END
```

Query

```
:MTESt:SSCReen:DPRinter:BACKground?
```

The query returns the current background for the printer.

Returned Format

```
[:MTESt:SSCReen:DPRinter:BACKground] {WHITe | NORMal} <NL>
```

Example

The following example gets the current background color setting and prints the setting on the controller display.

```
10  DIM SCREEN_BACK$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:BACKGROUND?"
30  ENTER 707;SCREEN_BACK$
40  PRINT SCREEN_BACK$
50  END
```

SSCReen:DPRinter:MEDia

Command

```
:MTESt:SSCReen:DPRinter:MEDia {PAPer |
TRANsparency}
```

The MEDia command specifies either paper or transparency in the printer. If TRANsparency is selected, the printer makes two passes over each data row, putting more ink on the page. This darkens the page for better transparency results. The command applies only to the HP PaintJet and Color DeskJet printers, and will slow print speed.

Example

The following program sets the media type to TRANsparency.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:MEDIA TRANSPARENCY"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:MEDia?
```

The query returns the current printer media.

Returned Format

```
[:MTESt:SSCReen:DPRinter:MEDia] {PAPer | TRANsparency}<NL>
```

Example

The following program gets the current media setting and prints the result on the controller screen.

```
10  DIM SCREEN_MEDIA$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:MEDIA?"
30  ENTER 707;SCREEN_MEDIA$
40  PRINT SCREEN_MEDIA$
50  END
```

SSCReen:DPRinter:PFORmat

Command

```
:MTESt:SSCReen:DPRinter:PFORmat {THINkjet |
PAINtjet | LASerjet | EPSon | DJ500 | DJ550 |
BWPaintjet | BWDeskjet}
```

The PFORmat (Printer FORmat) command selects the printer format to use when storing the screen to a printer. This includes only the formats available from the setup print menu that refer to a physical printer.

Example

The following program sets the save screen printer format to the HP DeskJet 550.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT DJ550"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:PFORmat?
```

The query returns the current printer format.

Returned Format

```
[:MTESt:SSCReen:DPRinter:PFORmat] {THINkjet | PAINtjet |
LASerjet | EPSon | DJ500 | DJ550 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example gets the current save screen printer format setting and prints it on the controller display.

```
10  DIM SCREEN_PFORMAT$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PFORMAT?"
30  ENTER 707;SCREEN_PFORMAT$
40  PRINT SCREEN_PFORMAT$
50  END
```

SSCReen:DPRinter:PORT

Command

```
:MTESt:SSCReen:DPRinter:PORT {CENTronics | HPIB}
```

The PORT command selects the printer port for the screen when store screen is going to the printer.

Example

The following example sets the Centronics port for printer output when the mask test saves a screen to the printer.

```
10  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT CENTRONICS"
20  END
```

Query

```
:MTESt:SSCReen:DPRinter:PORT?
```

The query returns the current port type.

Returned Format

```
[:MTESt:SSCReen:DPRinter:PORT]{CENTronics | HPIB} <NL>
```

Example

The following example determines which port is selected for save screen printer output and prints the result on the controller screen.

```
10  DIM SCREEN_PORT$[30]
20  OUTPUT 707;":MTEST:SSCREEN:DPRINTER:PORT?"
30  ENTER 707;SCREEN_PORT$
40  PRINT SCREEN_PORT$
50  END
```

SSUMmary

Command

```
:MTESt:SSUMmary {OFF | PRINter | DISK
[,<filename>]}
```

The :MTESt:SSUMmary (Store SUMmary) command saves the summary in the event of a failure.

A destination of OFF turns off the summary save.

A destination of PRINter builds the summary file and sends it to the printer. In this case a variety of printer-related controls are used to specify the printer configuration. The following printer commands are used when storing the summary to a printer:

- ADDRess,
- MEDia,
- PFORmat (printer format), and
- PORT.

When set to DISK, the summary is written to the disk drive using a user-specified base name with a machine generated suffix. The results of tests 1 2 and 3 for example may be written to files SUMM0001.SUM SUMM0002.SUM, and SUMM0003.SUM. The summary is a logging method where the user can get an overall view of the test results. The summary is an ASCII file which the user can peruse on a computer, read into a spreadsheet, and so on.

`<filename>` A file prefix of four characters defaulted to MSUM.

Example

The following program disables the save summary feature.

```
10  OUTPUT 707;":MTEST:SSUMMARY OFF"
20  END
```

Query

```
:MTESt:SSUMmary?
```

The query returns the current specified destination for the summary.

Returned Format

```
[:MTESt:SSUMmary] {OFF | PRINter | DISK [,<filename>]}<NL>
```

Example

The following example gets the setting of the save summary destination from the oscilloscope and prints the result on the controller display.

```
10  DIM SUMMARY_DEST$[30]
20  OUTPUT 707;":MTEST:SSUMMARY?"
30  ENTER 707;SUMMARY_DEST$
40  PRINT SUMMARY_DEST$
50  END
```

SSUMmary:ADDRess

Command

```
:MTESt:SSUMmary:ADDRess <address_value>
```

The :MTESt:SSUMmary:ADDRess command selects the HP-IB address for the printer. This address is used only when the summary is going to the printer and the port is set to HP-IB.

`<address_value>` Any HP-IB address from 0 to 30.

Example

The following program sets the printer output port to HP-IB at address 9.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PORT HPIB"
20  OUTPUT 707;":MTEST:SSUMMARY:ADDRESS 9"
30  END
```

Query

```
:MTESt:SSUMmary:ADDRess?
```

The query returns the current address of the printer.

Returned Format

```
[:MTESt:SSUMmary:ADDRess]<address_value><NL>
```

Example

The following example gets the current setting of the HP-IB address and prints it on the controller display.

```
10  DIM SUMMARY_ADDR$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:ADDRESS?"
30  ENTER 707;SUMMARY_ADDR$
40  PRINT SUMMARY_ADDR$
50  END
```

SSUMmary:MEDia

Command

```
:MTESt:SSUMmary:MEDia {PAPer | TRANsparency}
```

The :MTESt:SSUMmary:MEDia command specifies either paper or transparency in the printer. When TRANsparency is selected, the printer prints the data twice, which makes the contents of the paper look darker and slows down the printing process. This control is used when the store summary is directed to the printer, and is only valid for the HP PaintJet and Color DeskJet printers.

Example

The following program sets the media type to paper for the store summary results.

```
10  OUTPUT 707;":MTEST:SSUMMARY:MEDIA PAPER"
20  END
```

Query

```
:MTESt:SSUMmary:MEDia?
```

The query returns the current specified media for the printer.

Returned Format

```
[:MTESt:SSUMmary:MEDia]{PAPer | TRANsparency}<NL>
```

Example

The following example gets the media type setting and prints the result on the controller display.

```
10  DIM SUMMARY_MEDIA$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:MEDIA?"
30  ENTER 707;SUMMARY_MEDIA$
40  PRINT 707;SUMMARY_MEDIA$
50  END
```

SSUMmary:PFORmat

Command

```
:MTESt:SSUMmary:PFORmat {THINkjet | PAINtjet |
LASerjet | EPSon | DJ550 | BWPaintjet | BWDeskjet}
```

The :MTESt:SSUMmary:PFORmat (Print FORmat) command selects the printer format to use when storing the summary to a printer. This includes the physical printers supported by the instrument with the exception of the HP DeskJet 500C with the color cartridge installed. To set up the format for the HP DeskJet 500C, use the monochrome cartridge and the BWDeskjet parameter in the command.

Example

The following program sets the save screen printer format to Epson.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PFORMAT EPSON"
20  END
```

Query

```
:MTESt:SSUMmary:PFORmat?
```

The query returns the current specified printer format.

Returned Format

```
[:MTESt:SSUMmary:PFORmat] {THINkjet | PAINtjet | LASerjet |
EPSon | DJ550 | BWPaintjet | BWDeskjet}<NL>
```

Example

The following example gets the current printer format setting and prints it on the controller screen.

```
10  DIM SUMMARY_FORMAT$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:PFORMAT?"
30  ENTER 707;SUMMARY_FORMAT$
40  PRINT SUMMARY_FORMAT$
50  END
```

SSUMmary:PORT

Command

```
:MTESt:SSUMmary:PORT {CENTronics | HPIB}
```

The :MTESt:SSUMmary:PORT command selects the printer port for the summary when the summary is sent to the printer.

Example

The following program sets the printer port for summary output to HP-IB.

```
10  OUTPUT 707;":MTEST:SSUMMARY:PORT HPIB"
20  END
```

Query

```
:MTESt:SSUMmary:PORT?
```

The query returns the current specified port for the printer.

Returned Format

```
[:MTESt:SSUMmary:PORT]{CENTronics | HPIB} <NL>
```

Example

The following program gets the current setting of the summary output port and prints the setting on the controller display.

```
10  DIM SUMMARY_PORT$[30]
20  OUTPUT 707;":MTEST:SSUMMARY:PORT?"
30  ENTER 707;SUMMARY_PORT$
40  PRINT SUMMARY_PORT$
50  END
```

SWAVeform

Command

```
:MTESt:SWAVeform  <source>, <destination>,
[<filename>[,<format>]]
```

The :MTESt:SWAVeform (Store WAVeform) command saves waveforms from a channel, function, histogram, or waveform memory in the event of a failure detected by the Mask Test. Each waveform source can be individually specified allowing multiple channels or functions to be saved to disk or waveform memories. Setting a particular source to OFF removes any waveform save action from that source.

`<source>` {CHANnelN | FUNCtionN | FFT | HISTogram | WMEMoryN}

`<destination>` {OFF | WMEMoryN | DISK}

`<filename>` A descriptive file prefix consisting of up to four characters. If no filename is specified the prefix defaults to CH1A...CH4B, FN1, FN2, FFT, HIST, MEM1..MEM4.

`<format>` {TEXT [,YVALues> | VERBose] | INTernal}

Example

The following example saves the waveform from channel 1 to waveform memory 1 when a failure occurs.

```
10  OUTPUT 707;":MTEST:SWAVEFORM CHANNEL1,WMEMORY1"
20  END
```

Query

```
:MTESt:SWAVeform? <source>
```

The query returns the current state of the :MTESt:SWAVeform command.

Returned Format

```
[:MTESt:SWAVeform]<source>, <destination>,
[<filename>[,<format>]]<NL>
```

Example

The following example gets the current save waveform configuration and prints it on the controller display.

```
10  DIM SAVE_WAVE$[200]
20  OUTPUT 707;":MTEST:SWAVEFORM? CHANNEL1"
30  ENTER 707;SAVE_WAVE$
40  PRINT SAVE_WAVE$
50  END
```

TEST

Command

```
:MTESt:TEST {{ON | 1} {OFF | 0}}
```

The MTESt:TEST command controls the execution of the Mask Test function. On allows the Mask Test to run for all of the active sources. When the Mask Test is turned on the Mask Test results are displayed on screen in a window below the graticule in the mask test window. Off disables mask testing.

Example

The following example turns on the mask test function.

```
10  OUTPUT 707;":MTEST:TEST ON"
20  END
```

Query

```
:MTESt:TEST?
```

The query returns the state of the mask test subsystem, whether on or off.

Returned Format

```
[:MTESt:TEST] {1 | 0} <NL>
```

Example

The following example determines whether the mask test subsystem is on or off and prints the result on the controller screen.

```
10  DIM MTEST_STATE$[30]
20  OUTPUT 707;":MTEST:TEST?"
30  ENTER 707;MTEST_STATE$
40  PRINT MTEST_STATE$
50  END
```

29

Histogram Subsystem

Histogram Commands

The Histogram commands and queries control the histogram features of the HP 54720 oscilloscope. A histogram is a probability distribution that shows the distribution of acquired data within a user-definable histogram window. You can display the histogram either vertically for voltage measurements or horizontally for timing measurements.

The most common use for histograms is measuring and characterizing noise or jitter on displayed waveforms. Noise is measured by sizing the histogram window to a narrow portion of time and observing a vertical histogram that measures the noise on a waveform. Jitter is measured by sizing the histogram window to a narrow portion of voltage and observing a horizontal histogram that measures the jitter on an edge.

The Histogram subsystem contains the following commands:

- AXIS,
- MODE,
- RUNTil,
- SCALe,
- SCALe:OFFSet,
- SCALe:RANGe,
- SCALe:SCALe,
- SCALe:TYPE,
- WINDow:SOURce,
- WINDow:X1Position,
- WINDow:X2Position,
- WINDow:Y1Position, and
- WINDow:Y2Position.

Figure 29-1 is the syntax diagram for the Histogram subsystem commands.

Histograms and the Database

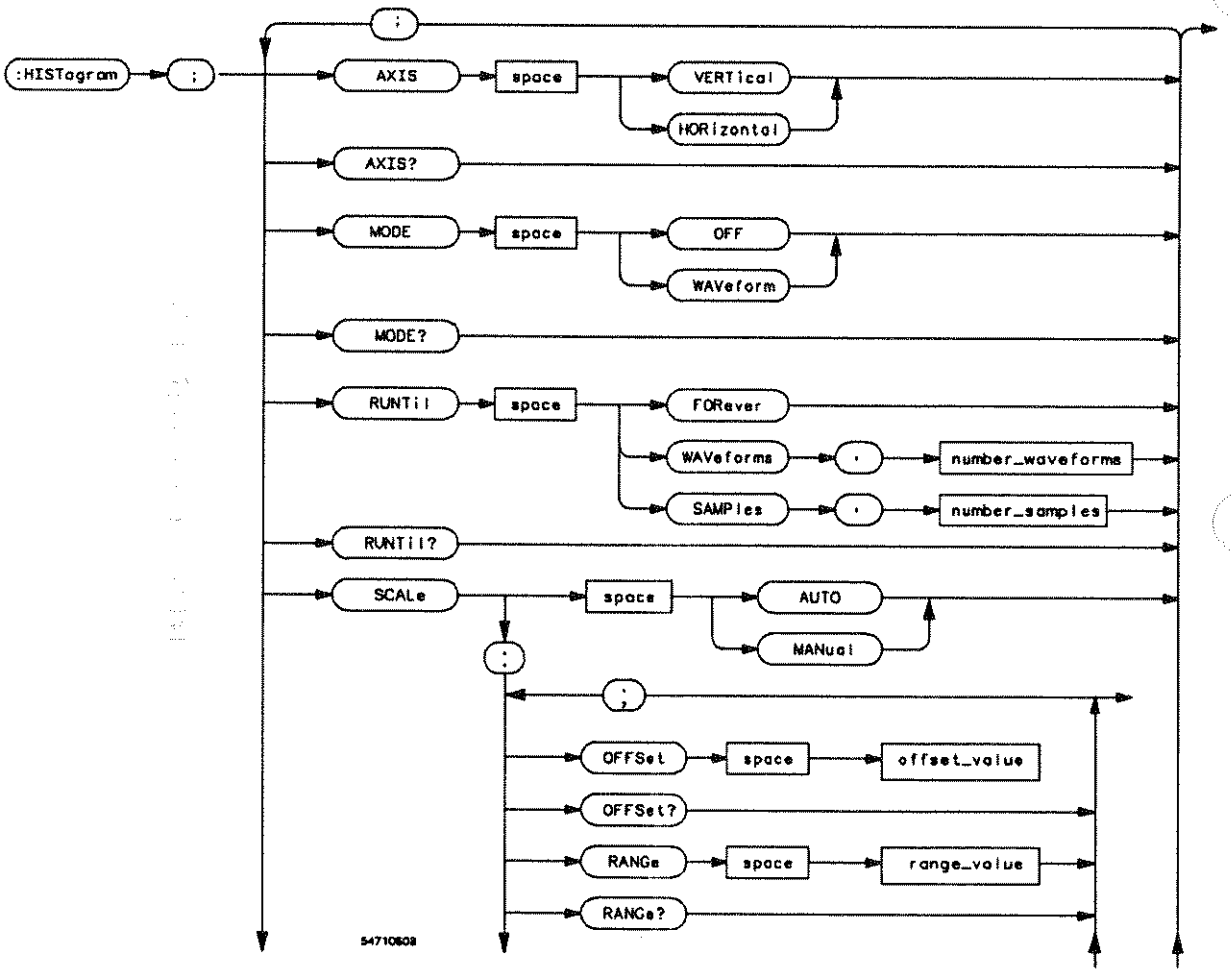
The histograms, mask testing, and color-graded display use a specific database in the oscilloscope that uses a different memory area from the waveform record for each channel. When any of these features are turned on, the oscilloscope starts building the database. The database is the size of the graticule area, which is 256 pixels high by 451 pixels wide. Behind each pixel is a 16-bit counter that is incremented each time data from a channel or function hits a pixel. The maximum count (saturation) for each counter is 63,488. You can use the DISPlay:CGRade command to see if any of the counters are close to saturation.

The database continues to build until the oscilloscope stops acquiring data or all three functions (color-graded display, mask testing, and histograms) are turned off. You can set the RUNTil (Run Until) mode to stop acquiring data after a specified number of waveforms or samples are acquired. You can clear the database by turning off all three features that use the database.

The database does not differentiate waveforms from different channels or functions. If three channels are turned on and the waveform from each channel happens to light the same pixel at the same time, the counter is incremented by three. However, it is not possible to tell how many hits came from each waveform. To separate waveforms, you can set the display to two graphs or position the waveforms vertically with the channel offset. By separating the waveforms, you can avoid overlapping data in the database caused by multiple waveforms. Even if the display is set to show only the most recent acquisition, the database keeps track of all pixel hits while the database is building.

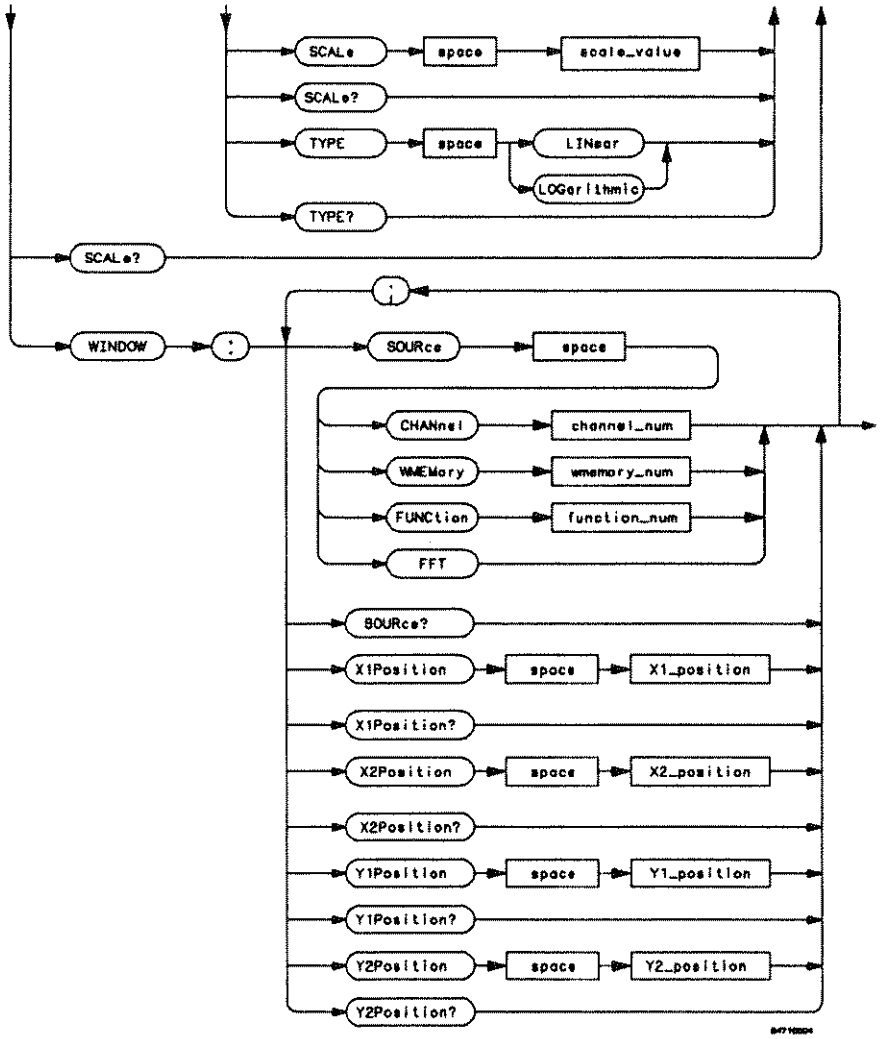
Remember that the color-graded display, mask testing, and histograms all use the same database. Suppose that the database is building because color-graded display is ON; when mask testing or histograms are turned on, they can use the information already established in the database as though they had been turned on the entire time.

Figure 29-1



Histogram Subsystem Syntax Diagram

Figure 29-1 (continued)



Histogram Subsystem Syntax Diagram (continued)

AXIS

Command

```
:HISTogram:AXIS {VERTical | HORizontal}
```

The :HISTogram:AXIS command selects the axis of the histogram. A horizontal or vertical histogram may be created.

Example

The following example defines a vertical histogram.

```
10  OUTPUT 707;":HISTOGRAM:AXIS VERTICAL"
20  END
```

Query

```
:HISTogram:AXIS?
```

The query returns the currently selected histogram axis.

Returned Format

```
[:HISTogram:AXIS] {VERTical | HORizontal} <NL>
```

Example

The following example returns the result of the axis query and prints it to the controller's screen.

```
10  DIM AXIS$[50]
20  OUTPUT 707;":HISTOGRAM:AXIS?"
30  ENTER 707;AXIS$
40  PRINT AXIS$
50  END
```

MODE

Command

```
:HISTogram:MODE {OFF | WAVeform}
```

The HISTogram:MODE command selects the histogram mode. The histogram may be off or set to track the waveform database.

Example

The following example sets the histogram mode to track the waveform database.

```
10  OUTPUT 707;":HISTOGRAM:MODE WAVEFORM"
20  END
```

Query

```
:HISTogram:MODE?
```

The query returns the currently selected histogram mode.

Returned Format

```
[:HISTogram:MODE] {OFF | WAVeform} <NL>
```

Example

The following example returns the result of the mode query and prints it to the controller's screen.

```
10  DIM MODE$[10]
20  OUTPUT 707;":HISTOGRAM:MODE?"
30  ENTER 707;MODE$
40  PRINT MODE$
50  END
```

RUNTil

Command

```
:HISTogram:RUNTil {FORever | {WAVeforms, <n
waveforms>} | {SAMples, <n samples>}}
```

The :HISTogram:RUNTil command selects the histogram run until mode. The histogram may be set to run until n waveforms or n samples have been acquired or will run forever. If more than one run until criteria is set, then the instrument will act upon the completion of run until criteria which is achieved first.

Example

The following example specifies that the histogram runs until 200 samples have been obtained.

```
10  OUTPUT 707;":HISTOGRAM:RUNTIL SAMPLES,200"
20  END
```

Query

```
:HISTogram:RUNTil?
```

The query returns the currently selected run until state.

Returned Format

```
[:HISTogram:RUNTil] {FORever | {WAVeform, <n waveforms>} |
{SAMples, <n samples>}}  <NL>
```

Example

The following example returns the result of the run until query and prints it to the controller's screen.

```
10  DIM RUNT$[50]
20  OUTPUT 707;":HISTOGRAM:RUNTIL?"
30  ENTER 707;RUNT$
40  PRINT RUNT$
50  END
```

SCALe

Command

```
:HISTogram:SCALe  {AUTO | MANual}
```

The :HISTogram:SCALe command selects the histogram scaling mode. The scaling mode may be automatic or manual. In automatic scaling, the scale will be set to display the histogram using one half of the display. In manual scaling, the scale and offset may be set manually to show any portion of the histogram.

Example

The following example selects automatic scaling.

```
10  OUTPUT 707;":HISTOGRAM:SCALE AUTO"
20  END
```

Query

```
:HISTogram:SCALe?
```

The query returns the currently selected histogram scale mode.

Returned Format

```
[:HISTogram:SCALe]{AUTO | MANual} <NL>
```

Example

The following example returns the result of the scale query and prints it to the controller's screen.

```
10  DIM SCAL$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE?"
30  ENTER 707;SCAL$
40  PRINT SCAL$
50  END
```

SCALe:OFFSet

Command

```
:HISTogram:SCALe:OFFSet {offset}
```

The :HISTogram:SCALe:OFFSet command sets the histogram offset. For horizontal histograms, this is the vertical offset in percentage of peak or decibels. For vertical histograms, this is the horizontal offset in percentage of peak or decibels.

For the linear scale type, the offset is the percentage of peak at the left or lower edge of the display. For example with a horizontal histogram, an offset of 50% would place one half of the peak (50%) at the lower edge of the display. 50% of the histogram would be below the display and the other 50% would be above the lower edge of the display.

For the log scale type, the offset is in decibels at the left or lower edge of the display. The histogram is plotted according to the formula:

$y = 20 * log\ 10\ (x/peak)$

where x is the number of hits of a histogram column and *peak* is the number of hits of the largest histogram column. This means that 0 db is at the histogram peak and that the offset can only contain negative values. For example, with a horizontal histogram, an offset of -20 db would place 10% or $10^{(-20/20)}$ of the peak at the lower edge of the display.

The histogram scale should be in manual mode when the offset is modified.

Example

The following example sets the offset to 50%.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:OFFSET 50"
20  END
```

Query

```
:HISTogram:SCALe:OFFSet?
```

The query returns the current offset in decibels or percentage of hits.

Returned Format

```
[:HISTogram:SCALe:OFFSet] {offset} <NL>
```

Example

The following example returns the result of the offset query and prints it to the controller's screen.

```
10  DIM OFF$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:OFFSET?"
30  ENTER 707;OFF$
40  PRINT OFF$
50  END
```

SCALe:RANGe

Command

```
:HISTogram:SCALe:RANGe {range}
```

The :HISTogram:SCALe:RANGe command selects the histogram scale across the entire display. For horizontal histograms, this is the vertical percentage of peak or decibels across the display. For vertical histograms, this is the horizontal percentage of peak or decibels across the display.

For the linear scale type, the range is in the percentage of the peak across the display. For example with a horizontal histogram, 200% would place 25% of the histogram in each of 8 divisions with the top of the peak (100%) at the middle of the display.

For the log scale type, the range is in decibels full scale. The histogram is plotted according to the formula:

$y = 20 * \log_{10}(x/peak)$

where x is the number of hits of a histogram column and *peak* is the number of hits of the largest histogram column.

The histogram scale should be in manual mode when the range is modified.

Example

The following example sets the range of the histogram to 200%.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:RANGE 200"
20  END
```

Query

```
:HISTogram:SCALe:RANGe?
```

The query returns the current range in decibels or percentage of hits across the display.

Returned Format

```
[:HISTogram:SCALe:RANGe] {range}<NL>
```

Example

The following example returns the result of the range query and prints it to the controller's screen.

```
10  DIM RANGE$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:RANGE?"
30  ENTER 707;RANGE$
40  PRINT RANGE$
50  END
```

SCALe:SCALe

Command

```
:HISTogram:SCALe:SCALe {scale}
```

The :HISTogram:SCALe:SCALe command selects the histogram scale per division. For horizontal histograms, this is the vertical percentage of peak or decibels per division. For vertical histograms, this is the horizontal percentage of peak or decibels per division.

For the linear scale type, the scale is in the percentage of peak per division. For example with a horizontal histogram, 25% would place one quarter of the histogram in each of 8 divisions with the top of the peak (100%) at the middle of the display.

For the log scale type, the scale is in decibels per division. The histogram is plotted according to the formula:

$y = 20 * log\ 10\ (x/peak)$

where x is the number of hits of a histogram column and *peak* is the number of hits of the largest histogram column.

The histogram scale should be in manual mode when the scale is modified.

Example

The following example sets the histogram scale to 25% (per division).

```
10  OUTPUT 707;":HISTOGRAM:SCALE:SCALE 25"
20  END
```

Query

```
:HISTogram:SCALe:SCALe?
```

The query returns the current range in decibels or percentage of hits across the display.

Returned Format

```
[:HISTogram:SCALe:SCALe] {scale} <NL>
```

Example

The following example returns the result of the scale query and prints it to the controller's screen.

```
10  DIM SCSCALE$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:SCALE?"
30  ENTER 707;SCSCALE$
40  PRINT SCSCALE$
50  END
```

SCALe:TYPE

Command

```
:HISTogram:SCALe:TYPE {LINear | LOGarithmic}
```

The :HISTogram:SCALe:TYPE command selects the histogram scale type. The histogram may be displayed according to a linear or a logarithmic scale.

Example

The following example sets the histogram scale to linear.

```
10  OUTPUT 707;":HISTOGRAM:SCALE:TYPE:LINEAR"
20  END
```

Query

```
:HISTogram:SCALe:TYPE?
```

The query returns the currently selected histogram scale type.

Returned Format

```
[:HISTogram:SCALe:TYPE] {LINear | LOGarithmic} <NL>
```

Example

The following example returns the result of the scale type query and prints it to the controller's screen.

```
10  DIM TYPE$[50]
20  OUTPUT 707;":HISTOGRAM:SCALE:TYPE?"
30  ENTER 707;TYPE$
40  PRINT TYPE$
50  END
```

WINDow:SOURce

Command

```
:HISTogram:WINDow:SOURce {CHANnelN | FUNCtionN |
WMEMoryN | FFT}
```

The :HISTogram:WINDow:SOURce command selects the source of the histogram window. The histogram window will track the source's vertical and horizontal scale.

Example

The following example sets the histogram window's source to CHANNEL1.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:SOURCE CHANNEL1"
20  END
```

Query

```
:HISTogram:WINDow:SOURce?
```

The query returns the currently selected histogram window source.

Returned Format

```
[:HISTogram:WINDow:SOURce] {CHANnelN | FUNCtionN | WMEMoryN |
FFT }<NL>
```

Example

The following example returns the result of the window source query and prints it to the controller's screen.

```
10  DIM WINSOUR$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:SOURCE?"
30  ENTER 707;WINSOUR$
40  PRINT WINSOUR$
50  END
```

WINDow:X1Position

Command

```
:HISTogram:WINDow:X1Position {X1 position}
```

The :HISTogram:WINDow:X1Position command moves the X1 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example

The following example sets the X1 position to -200 micro seconds.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:X1POSITION -200E-6"
20  END
```

Query

```
:HISTogram:WINDow:X1Position?
```

The query returns the value of the X1 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:X1Position]  {X1 position} <NL>
```

Example

The following example returns the result of the X1 position query and prints it to the controller's screen.

```
10  DIM X1$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:X1POSITION?"
30  ENTER 707;X1$
40  PRINT X1$
50  END
```

WINDow:X2Position

Command

```
:HISTogram:WINDow:X2Position {X2 position}
```

The :HISTogram:WINDow:X2Position command moves the X2 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example

The following example sets the X2 marker to 200 micro seconds.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:X2POSITION 200E-6"
20  END
```

Query

```
:HISTogram:WINDow:X2Position?
```

The query returns the value of the X2 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:X2Position] {X2 position} <NL>
```

Example

The following example returns the result of the X2 position query and prints it to the controller's screen.

```
10  DIM X2$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:X2POSITION?"
30  ENTER 707;X2$
40  PRINT X2$
50  END
```

WINDow:Y1Position

Command

```
:HISTogram:WINDow:Y1Position {Y1 position}
```

The :HISTogram:WINDow:Y1Position command moves the Y1 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the histogram window source.

Example

The following example sets the position of the Y1 marker to -250 millis.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:Y1POSITION -250E-3"
20  END
```

Query

```
:HISTogram:WINDow:Y1Position?
```

The query returns the value of the Y1 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:Y1Position] {Y1 position}<NL>
```

Example

The following example returns the result of the Y1 position query and prints it to the controller's screen.

```
10  DIM Y1$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:Y1POSITION?"
30  ENTER 707;Y1$
40  PRINT Y1$
50  END
```

WINDow:Y2Position

Command

```
:HISTogram:WINDow:Y2Position {Y2 position}
```

The :HISTogram:WINDow:Y2Position command moves the Y2 marker of the histogram window. The histogram window selects a portion of the database to histogram. The histogram window markers will track the scale of the historgram window source.

Example

The following example sets the position of the Y2 marker to 1.

```
10  OUTPUT 707;":HISTOGRAM:WINDOW:Y2POSITION 1"
20  END
```

Query

```
:HISTogram:WINDow:Y2Position?
```

The query returns the value of the Y2 histogram window marker.

Returned Format

```
[:HISTogram:WINDow:Y2Position] {Y2 position}<NL>
```

Example

The following example returns the result of the Y2 position query and prints it to the controller's screen.

```
10  DIM Y2$[50]
20  OUTPUT 707;":HISTOGRAM:WINDOW:Y2POSITION?"
30  ENTER 707;Y2$
40  PRINT Y2$
50  END
```

30

Error Messages

Error Messages

This chapter describes the error messages and how they are generated. The possible causes for the generation of the error messages are also listed in table 30-1.

Error Queue

As errors are detected, they are placed in an error queue. This queue is first in, first out. If the error queue overflows, the last error in the queue is replaced with error -350, "Queue overflow." Anytime the error queue overflows, the oldest errors remain in the queue, and the most recent error is discarded. The length of the oscilloscope's error queue is 30 (29 positions for the error messages, and 1 position for the "Queue overflow" message). Reading an error from the head of the queue removes that error from the queue, and opens a position at the tail of the queue for a new error. When all errors have been read from the queue, subsequent error queries return 0, "No error."

The error queue is cleared when any of the following occur:

- the instrument is powered up,
- a *CLS command is sent,
- the last item from the queue is read, or
- the instrument is switched from talk only to addressed mode on the front panel.

Error Numbers

The error numbers are grouped according to the type of error that is detected.

- +0 indicates no errors were detected.
- -100 to -199 indicates a command error was detected
- -200 to -299 indicates an execution error was detected.
- -300 to -399 indicates a device-specific error was detected.
- -400 to -499 indicates a query error was detected.
- +1 to +32767 indicates an oscilloscope specific error has been detected.

Command Error

An error number in the range -100 to -199 indicates that an IEEE 488.2 syntax error has been detected by the instrument's parser. The occurrence of any error in this class sets the command error bit (bit 5) in the event status register and indicates that one of the following events occurred:

- An IEEE 488.2 syntax error was detected by the parser. That is, a controller-to-oscilloscope message was received that is in violation of the IEEE 488.2 standard. This may be a data element that violates the oscilloscope's listening formats, or a data type that is unacceptable to the oscilloscope.
- An unrecognized header was received. Unrecognized headers include incorrect oscilloscope-specific headers and incorrect or unimplemented IEEE 488.2 common commands.
- A Group Execute Trigger (GET) was entered into the input buffer inside of an IEEE 488.2 program message.

Events that generate command errors do not generate execution errors, oscilloscope-specific errors, or query errors.

Execution Error

An error number in the range -200 to -299 indicates that an error was detected by the instrument's execution control block. The occurrence of any error in this class causes the execution error bit (bit 4) in the event status register to be set. It also indicates that one of the following events occurred:

- The program data following a header is outside the legal input range or is inconsistent with the oscilloscope's capabilities.
- A valid program message could not be properly executed due to some oscilloscope condition.

Execution errors are reported by the oscilloscope after expressions are evaluated and rounding operations are completed. For example, rounding a numeric data element will not be reported as an execution error. Events that generate execution errors do not generate command errors, oscilloscope specific errors, or query errors.

Device- or Oscilloscope-Specific Error

An error number in the range of -300 to -399 or +1 to +32767 indicates that the instrument has detected an error caused by an oscilloscope operation that did not properly complete. This may be due to an abnormal hardware or firmware condition. For example, this error may be generated by a self-test response error, or a full error queue. The occurrence of any error in this class causes the oscilloscope-specific error bit (bit 3) in the event status register to be set.

Query Error

An error number in the range -400 to -499 indicates that the output queue control of the instrument has detected a problem with the message exchange protocol. An occurrence of any error in this class should cause the query error bit (bit 2) in the event status register to be set. An occurrence of an error also means one of the following is true:

- An attempt is being made to read data from the output queue when no output is either present or pending.
- Data in the output queue has been lost.

List of Error Messages

Table 30-1 is a list of the error messages that are returned by the parser on this oscilloscope.

Table 30-1

Error Messages

Error Number	Error Message	Cause
0	No error	The error queue is empty. Every error in the queue has been read (SYSTEM:ERROR? query) or the queue was cleared by power-up or *CLS.
-100	Command error	This is the generic syntax error used if the oscilloscope cannot detect more specific errors.
-101	Invalid character	A syntactic element contains a character that is invalid for that type.
-102	Syntax error	An unrecognized command or data type was encountered.
-103	Invalid separator	The parser was expecting a separator and encountered an illegal character.
-104	Data type error	The parser recognized a data element different than one allowed. For example, numeric or string data was expected but block data was received.
-105	GET not allowed	A Group Execute Trigger was received within a program message.
-108	Parameter not allowed	More parameters were received than expected for the header.
-109	Missing parameter	Fewer parameters were received than required for the header.
-112	Program mnemonic too long	The header or character data element contains more than twelve characters.
-113	Undefined header	The header is syntactically correct, but it is undefined for the oscilloscope. For example, *XYZ is not defined for the oscilloscope.
-121	Invalid character in number	An invalid character for the data type being parsed was encountered. For example, a "9" in octal data.
-123	Numeric overflow	Number is too large or too small to be represented internally.
-124	Too many digits	The mantissa of a decimal numeric data element contained more than 255 digits excluding leading zeros.
-128	Numeric data not allowed	A legal numeric data element was received, but the oscilloscope does not accept one in this position for the header.

Error Number	Error Message	Cause
-131	Invalid suffix	The suffix does not follow the syntax described in IEEE 488.2 or the suffix is inappropriate for the oscilloscope.
-138	Suffix not allowed	A suffix was encountered after a numeric element that does not allow suffixes.
-141	Invalid character data	Either the character data element contains an invalid character or the particular element received is not valid for the header.
-144	Character data too long	
-148	Character data not allowed	A legal character data element was encountered where prohibited by the oscilloscope.
-150	String data error	This error can be generated when parsing a string data element. This particular error message is used if the oscilloscope cannot detect a more specific error.
-151	Invalid string data	A string data element was expected, but was invalid for some reason. For example, an END message was received before the terminal quote character.
-158	String data not allowed	A string data element was encountered but was not allowed by the oscilloscope at this point in parsing.
-160	Block data error	This error can be generated when parsing a block data element. This particular error message is used if the oscilloscope cannot detect a more specific error.
-161	Invalid block data	
-168	Block data not allowed	A legal block data element was encountered but was not allowed by the oscilloscope at this point in parsing.
-170	Expression error	This error can be generated when parsing an expression data element. It is used if the oscilloscope cannot detect a more specific error.
-171	Invalid expression	
-178	Expression data not allowed	Expression data was encountered but was not allowed by the oscilloscope at this point in parsing.
-200	Execution error	This is a generic syntax error which is used if the oscilloscope cannot detect more specific errors.
-222	Data out of range	Indicates that a legal program data element was parsed but could not be executed because the interpreted value is outside the legal range defined by the oscilloscope.

Error Number	Error Message	Cause
-223	Too much data	Indicates that a legal program data element of block, expression, or string type was received that contained more data than the oscilloscope could handle due to memory or related oscilloscope-specific requirements.
-241	Hardware missing	
-256	File name not found	
-310	System error	Indicates that a system error occurred.
-350	Queue overflow	Indicates that there is no room in the error queue and an error occurred but was not recorded.
-400	Query error	This is the generic query error.
-410	Query INTERRUPTED	
-420	Query UNTERMINATED	
-430	Query DEADLOCKED	
-440	Query UNTERMINATED after indefinite response	

31

Algorithms

Algorithms

One of the primary features of this oscilloscope is its ability to make automatic measurements on displayed waveforms. The information on algorithms is contained in the HP 54700 FrontPanel Reference. That reference provides details on how automatic measurement algorithms are calculated and offers some tips on how to improve results.

Index